National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

NL-339 (r. 88/04) c

Canada

University of Alberta

Autonomous Indoor Navigation of a Robot

by

© Murat Gökşin Bakır

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Master of Science

Department of Computing Science

Edmonton, Alberta
Fall 1990

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-65120-2

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Murat Gökşin Bakır

TITLE OF THESIS: Autonomous Indoor Navigation of a Robot

DEGREE: Master of Science

YEAR THIS DEGREE GRANTED: 1990

Permission is hereby granted to UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(Signed) Murat Bakır . . . . . .

Permanent Address:
Mustafa Kemal Paşa Sokak
56/4
İnebolu 37500
TURKEY

Date: 12/September/1990

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **Autonomous Indoor Navigation of a Robot** submitted by **Murat Gökşin Bakır** in partial fulfillment of the requirements for the degree of Master of Science.

Hong Zhang
(Supervisor)

Roger W Toogood

Wm. Armstrong

Xiaobo Li

Date: 7/September/1990

To
My
Parents

Anneme
ve
Babama

Abstract

The design and implementation of an indoor navigation system of an autonomous mobile robot requires robust estimation of its position and generation of safe trajectories. In this thesis we present a general localization method and a trajectory generation algorithm so that the robot can autonomously navigate indoors in a known environment. The localization methods introduced so far make use of sensor data usually too excessive for real-time processing. We exploit optical wheel encoders and sonar proximity sensors in order to localize the robot dynamically without disturbing the continuity of its motion. The encoder localization is based on observing differential changes in wheel positions, while the sonar localization algorithm extracts position information from sonar reflections. The sonar sensory system correlates the range data and the line segments constructed from the range data with an a priori environment map to obtain a position estimate. A general fusion algorithm for multiple sensory systems combines sensory position information on the basis of an explicit representation of the uncertainties of the various sensors. The trajectory generation algorithm has been developed on the basis of previously introduced methods. The algorithm provides continuous and smooth motion, and facilitates adaptivity to sensor feedback. The above methods are implemented on a physical mobile robot and the experimental results of the realized system are presented.

Acknowledgements

I would like to thank my supervisor, Hong Zhang, for his guidance, constructive criticism and support throughout this work and especially for teaching me the process of research and its accomplishment. I would also like to thank Roger Toogood for his advice and suggestions about the project, I extend this thanks to all members of the Robotics Research Group at University of Alberta.

A special thanks goes to Feral and Cengiz, for providing me shelter, food and loving care during the last weeks of my thesis. Thanks to all those people who shared their time, thoughts and feelings with me. Finally, I would like to thank Şeref Uğuriş, for his existence in this world.

Contents

1 Introduction 1

1.1 Thesis Goals . . . 3

1.2 Autonomous Mobile Robot Navigation . . . 4

1.2.1 Perception . . . 5

1.2.2 Control . . . 6

1.3 Previous Work . . . 7

1.4 Organization of the Thesis . . . 8

2 System Description 10

2.1 System Architecture . . . 10

2.1.1 The Robot . . . 10

2.1.2 Proximity Subsystem . . . 12

2.2 Software Organization . . . 14

2.2.1 The World Model . . . 15

3 Localization of the Robot Using Multiple Sensors 18

3.1 Background . . . 18

3.2 Sensor Uncertainty Modeling . . . 20

3.3 Consistency Test of a Single Sonar Reading . . . 21

3.4 Perception Models . . . 25

3.4.1 Monitoring the Motion by Dead-Reckoning 28
3.4.2 Monitoring the Motion by Sonar Sensors 29
3.5 Orientation Determination 31
3.6 Final Fusion of the Data 32
3.7 Summary 36

4 Trajectory Planning and Execution 37
4.1 Introduction 37
4.2 Motion Planning of Mobile Robots 38
4.3 Kinematics of the TRC Robot 41
4.4 Trajectory Generation 45
4.4.1 Basic Formulation of the Solution 45
4.4.2 Determination of the Solution Degree and Exit Distance Selection 48
4.5 Motion Control 53
4.5.1 Setting Wheel Velocities 53
4.5.2 Execution of the Trajectory 54
4.6 Summary 56

5 Implementation 58
5.1 Communication 59
5.1.1 Scheduler 60
5.2 Motion and Sensory Control 62
5.2.1 Motion 62
5.2.2 Exception handling 64
5.2.3 Sensor Control 65
5.3 Localization 66
5.3.1 Sonar Sensor Localization 66

5.3.2 Encoder Localization 70
5.3.3 Data Fusion 72
5.4 Trajectory Generation and Control 73
5.5 Experimental Results 76
5.6 Summary 79

6 Conclusion 94

6.1 Summary Discussion 94
6.2 Future Research 95

Bibliography 98

List of Figures

1.1 Wheel structures for mobile robots 2

2.1 The robot base 11
2.2 Control architecture of the robot 11
2.3 The communication scheme 12
2.4 Control Structure of the proximity subsystem 13
2.5 Sonar sampling in the hallway 14
2.6 Sonar array configuration 15
2.7 The framework of the navigation system 16
2.8 The world model 17
2.9 Representation of a part of the corridor 17

3.1 Normalization of sensor data 23
3.2 Sensor feedback 26
3.3 A multiple reflection pattern and its interpretation 27
3.4 Cone of reflection 28
3.5 Visible space 29
3.6 Nearness Test 30
3.7 Geometry of a reflection 31
3.8 A sample weighted least square fit 33
3.9 Detecting a deviation from the estimated path 34

3.10 Fitting curves to a set of data points 35

4.1 A sample trajectory for the mobile robot 38
4.2 The TRC in World Coordinates 42
4.3 The Turn Motion 44
4.4 Dividing the goal into sub-goals 46
4.5 Segments of a trajectory 47
4.6 Alternative solutions for trajectory generation 48
4.7 Two different solutions for the same trajectory 49
4.8 Avoiding collision by changing the exit distance 50
4.9 Destination constraint on the exit distance 51
4.10 Details of a curvature 52
4.11 Possible types of motion 54
4.12 A sample motion and its corresponding velocity profile 55
4.13 Updating a trajectory during navigation 57

5.1 The communication between processes 60
5.2 The interrupt scheme 63
5.3 The scheduling mechanism 65
5.4 The world map 68
5.5 Associating a point with a line 69
5.6 The search algorithm 69
5.7 Forming lines by point transformation 70
5.8 Encoder geometry 71
5.9 Data flow of localization 73
5.10 A sample run within the corridors 74
5.11 Correction of a trajectory by overriding the last two nodes 75

5.12 Real and estimated path 78
5.13 Localization errors along the path 80
5.14 x coordinate of the actual location and the sonar localization 82
5.15 x coordinate of the actual location and the encoder localization . . . 83
5.16 x coordinate of the actual and the estimated location 84
5.17 y coordinate of the actual location and the sonar localization 85
5.18 y coordinate of the actual location and the encoder localization . . . 86
5.19 y coordinate of the actual and the estimated location 87
5.20 Error in Encoder localization and final estimation in x coordinates . 88
5.21 Error in Sonar localization and final estimation in x coordinates . . . 89
5.22 Error in Encoder localization and final estimation in y coordinates . 90
5.23 Error in Sonar localization and final estimation in y coordinates . . . 91
5.24 Error in Encoder localization and final estimation in orientation . . . 92
5.25 Error in Sonar localization and final estimation in orientation 93

Chapter 1

Introduction

Most of today's industrial purpose robots are used in stationary applications. However, there are a several tasks for which a high degree of mobility is required. Production techniques can be significantly improved by using autonomous mobile robots, which can move independently in the manufacturing domain and transport materials and tools. Utilizing an autonomous mobile robot need not be limited to the manufacturing industry. Space exploration, underwater exploration, and service industry are some of the areas in which mobile robotics research finds wide applications.

By definition, a mobile robot possesses a means of locomotion. Two main types of terrain locomotion techniques used today are *legged* and *wheeled* locomotion. A legged vehicle is extremely difficult to control. The robot not only has to maintain its balance all the time, but also has to synchronize its legs. A considerable amount of research is being done in this area. Examples are biped machines with and without joints [Sugano, 1985], hopping machines [Raibert, 1984], quadroped machines with crab-like motions [Hirose, 1984], and hexaped machines [Zhimin and Dongying, 1985]. Legged locomotion assumes static stability. Static stability is easy to achieve as it requires relatively simple analysis and reasonable computational complexity. Dynamic stability, however, still requires substantial re-

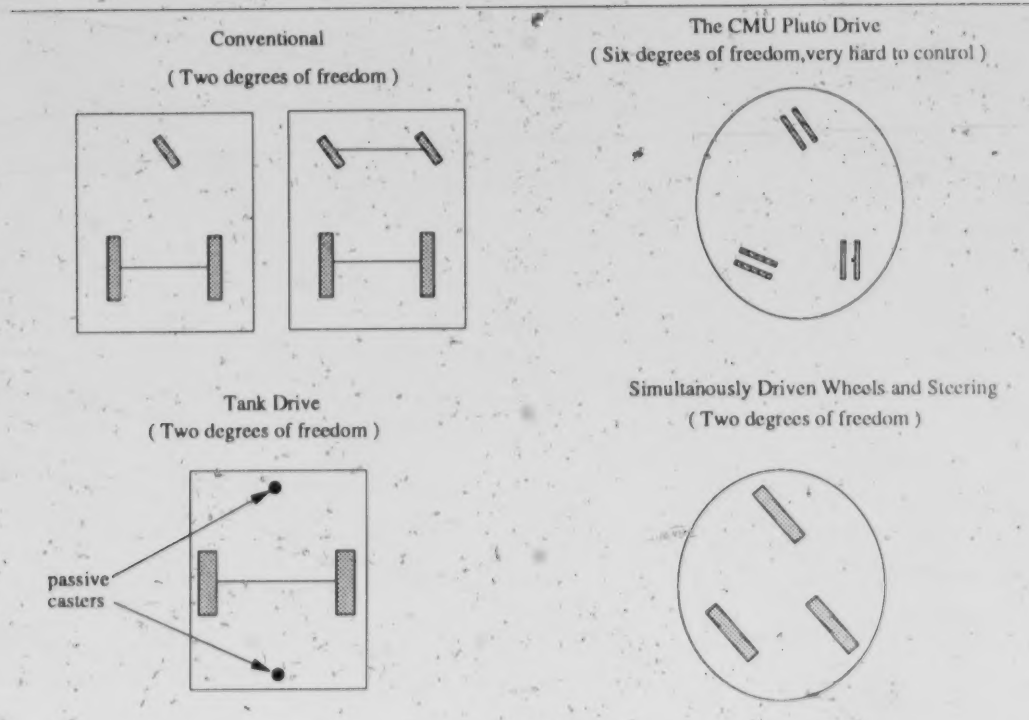
Conventional
(Two degrees of freedom)
The CMU Pluto Drive
(Six degrees of freedom,very hard to control)
Tank Drive
(Two degrees of freedom)
passive
casters
Simultanously Driven Wheels and Steering
(Two degrees of freedom)

Figure 1.1: Wheel structures for mobile robots

search.

Wheeled robots represent another class of locomotion. There is a large set of different designs for wheel structures, some of which are shown in Figure 1.1. A wheeled mobile robot locally moves in a plane, and thus it has three degrees of configurational freedom, which are the two translational degrees of freedom and one rotational degree of freedom on the $x - y$ plane. Depending on the kinematics of the robot, one of the configuration parameters may be dependent on the others. For example a tank drive mobile robot (Figure 1.1) can move only in the direction normal to the axis of its driving wheels. Wheeled robots with such constraints are referred to as *non-holonomic* robots.

Autonomous navigation requires the knowledge of the current state of the en-

vironment. This information can be acquired and updated by a number of individual sensors making use of various physical properties of the environment. A multitude of sensors have been developed [Brooks and Lozano-Perez, 1983]. In robot navigation, typical sensors used for perception are: odometer, compass, vision, proximity and tactile sensors.

1.1 Thesis Goals

This thesis presents a system for the autonomous indoor navigation of a wheeled mobile robot. The primary goal of this work is to provide a general framework under which positional information collected from various sensor sources can be fused to obtain an optimal estimate of the current position of the robot. We also present an algorithm that generates a continuous motion trajectory in a two-dimensional space according to the task specifications. In addition, unlike many previous attempts, we emphasize the real-time and sensor driven nature of the motion planning problem by updating the task plan on a continuous basis.

Estimating the position of a mobile robot is a critical issue in autonomous navigation as the success of any task that the robot has to perform relies on successful localization of the robot. A reliable localization method must use multiple sources so that the resulting position estimate will not be biased because of the deficiencies of a single type of sensor. The problem of localization requires a solution that employs a data integration method while considering the environmental and sensory characteristics that define the uncertainty related to each position estimate. The solution presented in this thesis exploits multiple sensors to dynamically localize the robot in a previously known environment without disturbing the continuity of its motion. Multiple sensor data are combined to obtain a final position estimate using a fusion method that employs probabilistic estimation methods.

Trajectory generation, on the other hand, is another problem to be addressed for navigation. Considering the fact that the robot will be realizing its tasks in real time, an efficient path generation algorithm is necessary. The method we develop generates safe trajectories which are formed of straight line segments and arcs of constant curvature, and maintains continuity of the motion with respect to the physical movement capabilities of the robot at hand.

The result of this study is a system that maintains its position while navigating in a previously known environment. The tasks to be executed are monitored dynamically by the supervisory control module in order to correct deviations from the path. This method provides a safe navigation within the environment. The theories we develop are verified physically on a non-holonomic mobile robot equipped with an ultrasonic sensor array and wheel encoders.

1.2 Autonomous Mobile Robot Navigation

Autonomy of a mobile robot is its ability to successfully navigate while performing its tasks without human supervision. The specific tasks, as well as the environmental constraints, define the level of autonomy that the robot must possess. For example, moving between two points in free space requires a lower level of autonomy than working in production tasks. However, apart from the environmental and task specific constraints, the control system of an autonomous mobile robot must contain at least two basic modules, *perception* and *control*. These two modules function independently and communicate and interact in real time. The perception system interprets the sensory data to obtain position estimates. The control system performs motion planning and execution.

Besides these requirements, an autonomous mobile robot may also be required to satisfy other conditions. Some of these can be examplified by object recognition for

a garbage collection task [Lin *et al.*, 1989], precise position estimation for an assembly task[Perez and Rouchy, 1987], etc. Our aim in realizing an autonomous mobile robot is to achieve a general methodology for autonomous navigation which will constitute the basis for future applications.

1.2.1 Perception

In order for a mobile robot to perform a navigation task, some means of environmental perception is required. Localization, in this context, can be defined as determining the current position estimate of the robot using sensors. There have been many solutions proposed for the problem of localization [Giralt *et al.*, 1987, Crowley, 1989b, Kuc and Barshan, 1989, Krotkov, 1989, Hu and Stockman, 1986, Kriegman *et al.*, 1987]. The basic idea of all these methods is to exploit one or more sensor systems to periodically estimate the position of the robot.

With the development of advanced sensor systems, like CCD cameras, laser and sonar rangefinders, etc., much information can be gathered from the environment. The usage of multiple sensors gives rise to the problem of combining data from these sensors. This problem is referred to as data fusion. Different sensors are perceptive to various aspects of the environment with different accuracies. Fusing the data at hand, considering all these constraints, is not an easy task. Probabilistic and heuristic methods are used to achieve this goal [Hackett and Shah, 1990], but the field is still in its infancy.

Our system interprets multiple sensory data into distinct position estimates based on the geometric characteristics of the sonar sensors and optical encoders mounted on robot wheels. The localization scheme uses a world model which represents the physical locations of the objects in the environment and has necessary information for the robot to perceive and differentiate between distinct objects and

their locations. All sonar data are associated with a certain amount of uncertainty and they are integrated into a single estimate using weighted least square fit. Encoder data is geometrically interpreted with a probabilistic uncertainty. Both of these sensory position estimates are fused into a final position estimate using maximum likelihood method. This process is repeated periodically to provide localization. The described fusion method is not limited to only sonar sensors and encoders, it is instead a general method for fusing sensor data from multiple sources.

1.2.2 Control

The basic task of the control system is to supervise the successful execution of continuous motion, such as reaching a final goal point, and to determine the deviations from the expected path. It is also responsible for decomposing a task into smaller subtasks with the help of decision making algorithms and problem solving techniques.

A reactive control plan is required to ensure the robustness of the system. The planning involves understanding a goal, and generating a feasible trajectory for it. The trajectory must be continuous, because discontinuities in the trajectory may result in infinite acceleration and deceleration. For indoor navigation, the geometric aspects of the environment must be perceived and an appropriate trajectory must be generated. There are two basic groups of methods for this problem: *local methods* and *global methods*. Local methods assume no knowledge about the environment and make use of local sensory information to generate partial trajectories [Khatib, 1986, Faverjon and Tournossoud, 1987]. Global methods, on the other hand, deal with cases where complete information about the navigation domain is provided [Laumond, 1987, Brooks and Lozano-Perez, 1983].

In this work we study both global and local trajectory generation. First, a global trajectory is generated for the main goal, and then, in order to carry it out,

it is decomposed into local trajectories. These local trajectories are monitored and dynamically updated to obtain a safe navigation based on sensory feedback from the perception system. A general trajectory generation method which uses straight line segments and arcs of constant curvature to maintain the continuity of motion is introduced. The motion planning accepts modifications in real time which are caused by possible positioning errors detected by the sensors.

1.3 Previous Work

The first project in mobile robot navigation appeared in 1967 as a thesis proposal [Schmidt, 1971]. The goal of the thesis was to realize a real time control of a computer driven vehicle which would make a trip from the house of the author to his work. However, the resulting system was not autonomous. Within the past two decades, autonomous navigation has been an active research area. Many autonomous navigation systems have been developed [Brooks, 1988, Crowley, 1989a, Stefik, 1985, Giralt, 1984, Kriegman *et al.*, 1987]. We will briefly review their hardware configurations in this section and will describe their perception and control structures relevant to our work in subsequent chapters.

The first automatically guided vehicles were mainly developed in the USA [Rembold and Levi, 1987] and Japan[Maeda, 1985]. Their characteristic features were sophisticated supervisory systems and onboard navigators. These vehicles were not completely autonomous. They were developed to operate on one factory floor with the help of induction loops, reflective guide strips, floor markings, etc.

DARPA [Stefik, 1985] has set the goal to build an autonomous land vehicle to be able to go on a mission over rugged terrains. Their vehicle is a six wheeled outdoor robot, equipped with 2 cameras and ultrasonic rangefinders. The planning activities are in four levels: mission planning, global trajectory planning, local planning,

and reflexive planning. Sensor processing is done in a hierarchical manner involving localization and reflexive reaction to the environment.

The Stanford mobile robot[Kriegman *et al.*, 1987] is a two wheeled vehicle, which has two cameras, sonar sensors and tactile sensors. The navigation system is both onboard (emergency handling) and offboard (global planning). Uncertainty is related to the position transformations by a multivariate normal distribution. The final information is used to update the global world model.

In France, the project HILARE [Giralt, 1984] began in 1974. Currently, it is an advanced testbed for mobile robot applications in LAAS. The locomotion system of HILARE involves 2 driving wheels and one caster wheel. The sensing hardware are 14 ultrasonic rangefinders and a combined camera/laser scanner for 3-D viewing. The control hierarchy of the robot is composed of planning, control analysis and navigation, and motion control. The world modeling and path finding take place in a hierarchical manner. Another mobile robot in France which uses laser and infrared triangulation sensors is VESA [Giralt, 1984]. Its locomotion mechanism uses 2 driving wheels and 2 passive casters. The control of VESA is done by a global planning module which determines the path by triangulation.

1.4 Organization of the Thesis

The following chapters describe an autonomous navigation system for a mobile robot. This system is implemented on a TRC mobile robot base [Tra, 1989b]. The hardware configuration and the control structure of the robot is described in chapter 2. In chapter 3 we introduce our solution to the problem of determining the optimal position estimate of a mobile robot using multiple sensors. This method is based on reducing the sensor uncertainty by data fusion. Chapter 4 describes a trajectory generation method, and the motion control of the robot to execute the generated path.

The implementation details of the proposed methods are described in chapter 5, which also gives the experimental results of the system. Finally, chapter 6 summarizes the work, and highlights directions for future research.

Chapter 2

System Description

In this chapter we will describe our autonomous mobile robot system, which explores the environment using an array of sonar proximity sensors to maintain its autonomy. The primary aim of our system is to dynamically maintain the location of the robot while controlling its motion towards a goal in a robust manner. The previous work in this field involves localization where the robot has to stop and investigate the environment [Elfes, 1986] [Crowley, 1985]. The system we introduce is sensor driven in real time where updates in positioning and motion planning are done on a continuous basis. The navigation, achieved by matching the extracted sensor information with the previously known world model, will be explained in the next chapters.

2.1 System Architecture

2.1.1 The Robot

The robot is constructed on a TRC mobile robot base [Tra, 1989b]. Its shape is

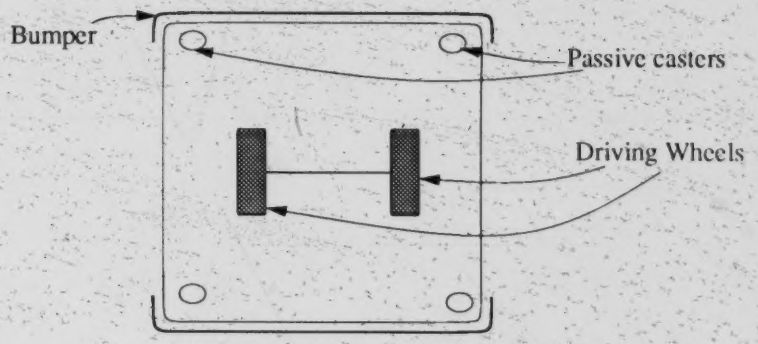

Figure 2.1: The robot base

rectangular with dimensions 28cm × 70cm × 75cm. Steering is done by 2 differential steering wheels on the center axis and 4 passive casters on the corners as shown in Figure 2.1. The independent control of the two wheels enable the robot to reach any point in its free space. This form of locomotion makes the vehicle well suited for indoor traveling.

The driver wheels' motors contain two shaft encoders used for odometry. The trajectories to be followed are specified by a path planner which will be explained in chapter 4.

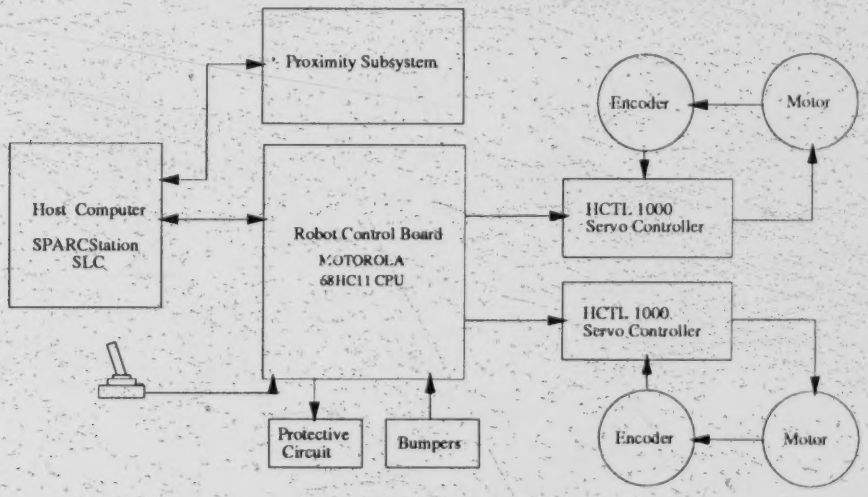

Figure 2.2: Control architecture of the robot

The robot is equipped with a Motorola 68HC11 CPU as the main controller. The safety bumpers at the two ends detect any contact with an obstacle and set certain registers for emergency handling. The software design of the controller allows linear motion and turning with a given radius and rate [Tra, 1989b] through multiple modes of motion such as *go* and *turn*. The transitions between these modes are practically continuous [Tra, 1989b]. The control architecture of the system is given in Figure 2.2. The robot's control system only communicates with the host system. Communication includes receiving motion commands from the host and sending the status of the robot to the host.

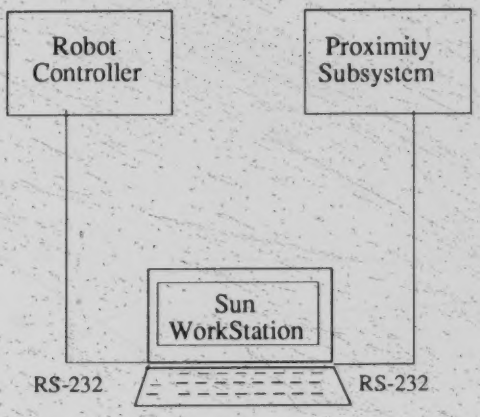

Figure 2.3: The communication scheme

The robot's computational system is currently an offboard Sun Workstation. As a host CPU, it communicates with the robot and the proximity subsystem via two RS232 serial ports. All the planning and perception is done on the host CPU. The resulting motion commands are sent to the robot and the sonar sensor controller. Figure 2.3 shows an overall system architecture.

2.1.2 Proximity Subsystem

The proximity subsystem is a sensor system capable of operating up to 24 sonar

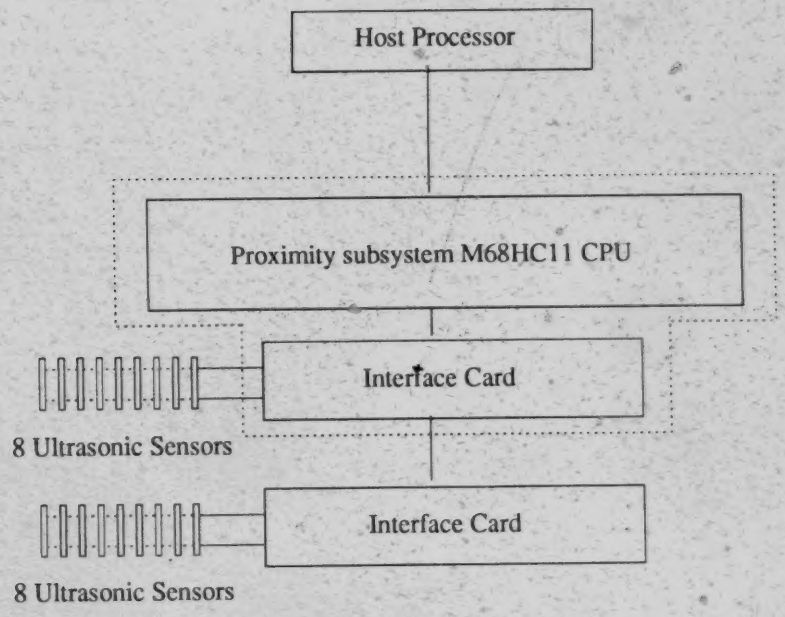

Figure 2.4: Control Structure of the proximity subsystem

and infrared sensors. Currently, in our system 16 sonar sensors are placed around the robot and provide direct range information. Experimentally each sonar sensor is found to be sensitive over a reception angle of approximately 30°. This result is consistent with other people's work [Moravec and Elfes, 1986]. The proximity subsystem also uses a 68HC11 processor to control 2 sensor interface boards, each of which can handle up to 8 sonar sensors (see Figure 2.4). The range of an ultrasonic sensor is between 15cm and 1050cm [Tra, 1989a]. The proximity software is able to activate sensors on a selective basis and set the timeout distance of each sensor. The major drawbacks of the system are its wide span angle and the long acoustic wavelength, which causes objects with smooth surfaces to reflect the wave speculatively and results in the robot's failure to detect those objects [Kriegman *et al.*, 1987]. Figure 2.5 illustrates the uncertainty in sonar sensor readings, showing the floor plan of the hallway where the experiments were run, and a sample scan of the corridor, using ultrasonic sensors. The sampling is done along a path starting from the middle of the corridor along the path shown with a dotted line in Figure 2.5.

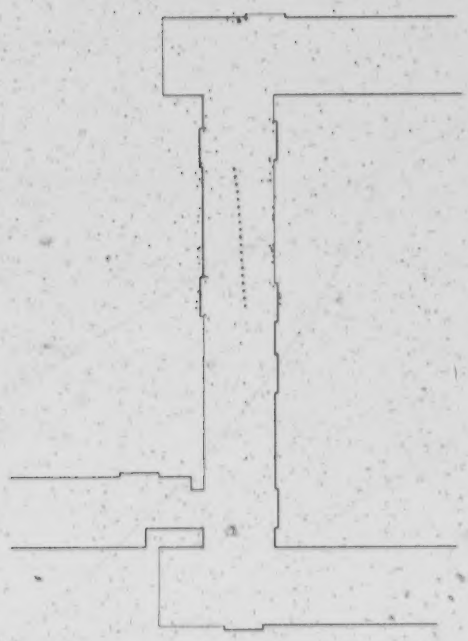

Figure 2.5: Sonar sampling in the hallway

The configuration of the sonar array of the robot is shown in Figure 2.6. The two sides and the front of the vehicle carry 3 sensors each. Two sensors oriented at 30° and 60°, respectively, are on each of the front corners, and one sensor oriented at 45° is located on each of the back corners. These sensors are used for the construction of an environment model and also for the detection of unexpected objects in order to react to them.

2.2 Software Organization

The control and perception organization of the robot is shown in Figure 2.7. The navigation system is based on two sensor systems, the encoders and the sonar sensors. The data coming from each of the sensor systems are processed independently to

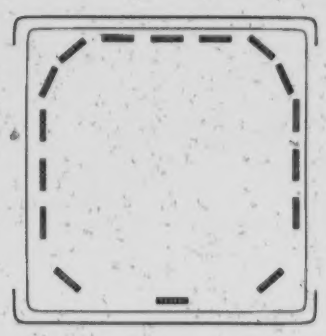

Figure 2.6: Sonar array configuration

localize the robot. Periodically, each sensor system produces an estimate of the robot position with a measure of uncertainty. The perception module receives input from both sensors and fuses them using a maximum likelihood estimator. The perception method involves matchings between the model and the a priori world model. These matchings determine the correspondence between the sensor and the world models. Then, based on these matchings and the uncertainties related to them, a new position for the robot is estimated. The final estimate of the position is sent to the trajectory planner. Using this current position estimate, the trajectory planner determines the deviations from the planned path, and updates the path planning, if necessary. The motion planner, which is the executor of the current trajectory, issues appropriate commands to the motion controller of the robot base in order to execute the planned trajectory. Details of the sensory perception and sensory fusion algorithms will be given in chapter 3. Further discussions of the trajectory generation will be provided in chapter 4.

2.2.1 The World Model

Designing the representation of the world model is a critical issue. The world model must describe the environment precisely without giving excessive details. The grid representation, where each cell in the map contains a probabilistic estimate

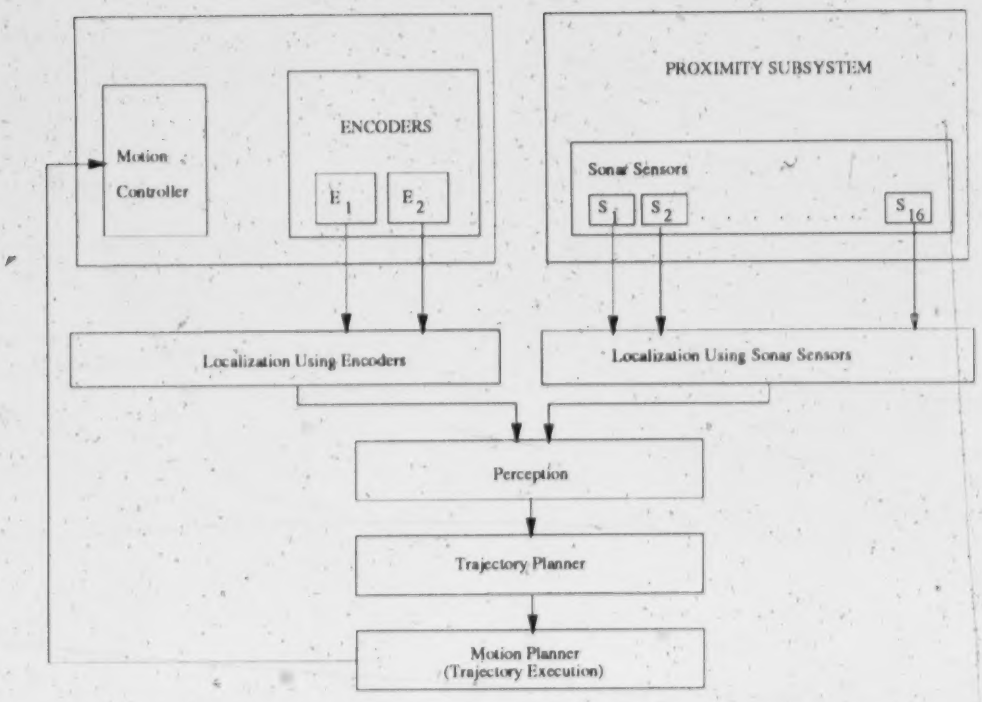

Figure 2.7: The framework of the navigation system

of whether it is empty or occupied by an object [Mathies and Elfes, 1988], is one alternative. However, it involves several geometric operations for any search through the map. Thus, a tree structured representation of the map is chosen. Since this model will be used for both the trajectory generation and the perception mechanisms, it should be kept as small as possible. Hierarchical representation is an alternative to a small map, as the search will be through a smaller map at each level of hierarchy.

Figure 2.8 shows the hierarchy of the world model within a building. Physically connected objects are also connected with a link at each level. Each node contains position information about itself. As the robot is designed to travel indoors on one floor only, the world model is represented in two dimensions. At the lowest level there are the line segments forming the structure of their parents. When searching for a path, the search will start at the current node and proceed by consecutive findpaths along successive levels of the map.

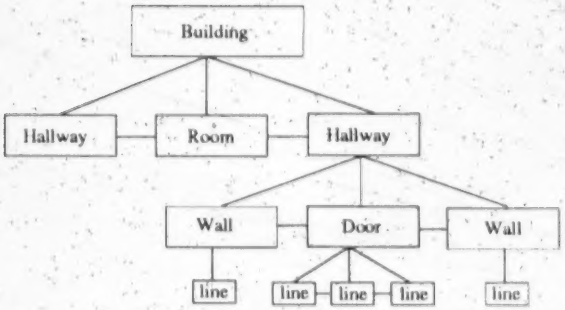

Figure 2.8: The world model

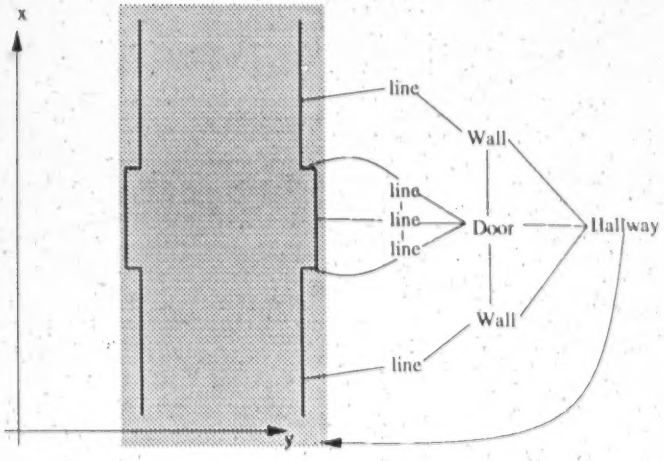

Figure 2.9: Representation of a part of the corridor

Let us go through a simple example to illustrate the representation of the world model. Figure 2.9 shows an example of the representation of a part of the corridor being modeled. The smallest unit of representation is a line. These lines are the projections of the surfaces (walls and doors) on the x-y plane. As seen in Figure 2.9, the lines which are physically connected at their edges are also connected in their abstract representations. At the higher levels of the hierarchy, a rough description is also given. In our example this description is the shaded area, which includes the hallway (and the rooms, if they are included). The advantage of this description is that one can find the region of a point at higher levels. Thus, generally a depth-first search is sufficient to find a particular point.

Chapter 3

Localization of the Robot Using Multiple Sensors

This chapter deals with the problem of determining the location and orientation of a mobile robot using multiple sensors, the process which is known as *localization*. The most commonly used technique for localization of a robot is dead reckoning (trajectory integration). However, because of the erratic nature of any sensor and many environmental constraints, such as a slippery floor, a dead-reckoning system will accumulate errors, and as a result, the robot will eventually lose track of its position along the path. To prevent this from happening, its position must be periodically re-established. Therefore, a solution for localization is essential for a mobile robot to perform a navigation task.

3.1 Background

There have been several solutions proposed for the problem of localization. The basic principle underlying these solutions is the fact that different sensors

are sensitive to different aspects of the environment. One type of sensor can be used in situations where another type does not function well, and vice-versa. The most common approach is to monitor the position of the robot using dead-reckoning. This can be done by monitoring wheel rotations using encoders attached to the wheels [Giralt *et al.*, 1987] [Tournossoud, 1988], or by detecting inertial forces [Thorpe and Kanade, 1986]. The main disadvantage of dead-reckoning is the fact that the errors caused by the terrain characteristics and encoder uncertainty will accumulate. Thus, the trajectory integration is not suitable for robust localization in the long run.

Using beacons at certain known positions to localize the robot is another approach to the problem. The basic idea is to make use of beacons, in the way lighthouses are used for ships. Some techniques involving this method are: using recognized landmarks [Levitt *et al.*, 1987], infrared position references [Giralt *et al.*, 1979], cable lines to follow [Drake *et al.*, 1985] and external lights as lighthouses [Lao *et al.*, 1986]. These methods involve measuring the distance to a number of beacons in order to find the relative position and orientation of the robot, and then mapping those into world coordinates. However, as the approach requires a very structured environment, it becomes expensive for applications which require a large work space.

Other approaches try to localize the robot in an unstructured environment. They can be grouped into two categories: those using active and those using passive sensors. Methods using active sensors are based on the principle of emitting a signal and interpreting the received signal, as a flying bat. Active sensors are widely used in autonomous navigation for proximity sensing. Ultrasonic sensors [Crowley, 1989b], [Miller, 1984], [Drumheller, 1987] can be used to measure the distances of objects in the environment to form an abstract world model to help localization. Other proximity sensors, like laser rangefinders, can also be used for this purpose [Smith and Cheeseman, 1987], [Hoppen *et al.*, 1990].Using sonar sensors is

quite inexpensive and useful for indoor navigation where the objects are not far away. However, because of the speed of sound in the air, the time limitations of the sonar sensors constitute a bottleneck. There are a few other problems for the sonar sensors. For example many objects are either absorbers or speculative reflectors [Kriegman *et al.*, 1987], and the sensors have a low angular resolution.

The second group of methods for localization is using passive sensors, which is usually done by vision. These methods make use of reflected light energy from the environment. Many techniques are proposed for position detection and world model construction [Kanade, 1987], [Crowley, 1989b] and [Krotkov, 1989] based on feature extraction and pattern matching. These methods are computationally complex. Restrictive assumptions can be made to decrease the complexity [Krotkov, 1989]. However, these assumptions can make the solutions unstable and error-prone.

All the methods described above make use of one or more sensor systems in addition to dead reckoning, in order to periodically localize the robot along its path. The advantages of using multiple sensors are straightforward. The uncertainty can be reduced by using a consensus of redundant sensors. Multiple data points on a certain geometry can be used to extract detailed information from the geometry. Redundancy can be used for detecting and compensating for sensor failures. Most of the implemented systems use consistency checking as the first step to eliminate false sensor data ([Lin *et al.*, 1989], [Brooks, 1988], [Crowley, 1989b], [Miller, 1984], [Drumheller, 1987] etc.). Consistency operators are applied to check sensor data and reject the sensor readings which do not fulfill the consistency criteria [Rodger and Browse, 1986]. Examples of decision criteria are: comparing the consistency of a sensor reading with the majority of the sensors [Lin *et al.*, 1989], and making use of physical characteristics of the sensors along with the knowledge of possible data sources [Crowley, 1989b] [Kriegman *et al.*, 1987]. In our case, the encoder readings are consistent throughout the process, but the sonar sensor readings may sometimes give inconsistent readings as in the example shown in Figure 3.3.

During self localization, as we get multiple observations about the environment from the sensors, the question of how to obtain an accurate description of the environment from these observations arises. Many answers have been proposed for this question based on either experimental heuristics or probabilistic calculations. Some of these methods can be grouped as: *Bayesian approaches, averaging and least square fitting* and *heuristic methods.* An example of Bayesian approaches may be using the Bayesian rule directly in a cellular representation of the environment, called occupancy grids [Mathies and Elfes, 1988], where each cell is associated with a probability of being occupied by an object. Another approach is to involve the maximum likelihood method to integrate the geometric sensor observations into an estimate of the environment, using uncertainties related to the sensors [Durrant-Whyte, 1987]. Averaging and least square methods are also used to match the features, extracted from the sensor readings, with the world model in order to estimate some unknown features of the environment [Kent *et al.*, 1986]. Both of the two methods use assumptions about the distribution characteristics of the sensor data. Heuristics (which are basically applying rules and relationships for model matching) are also widely used along with the other methods [Hu and Stockman, 1986]. Guiding, for example, involves focusing on one or more sensors within different parts of the scene [Luo *et al.*, 1988]. Heuristic methods are useful in isolated environments. However, they are the least adaptive methods [Hackett and Shah, 1990].

In this chapter we will describe the work we have done for the self localization of the TRC Mobile Robot. Our approach is using a Gaussian modeling for the sensor uncertainty. The localization of the robot is done by using the information from sonar sensors and encoders. Encoders keep track of the current position of the robot with a certain amount of uncertainty. The position information of the sonar perception is constructed by the robot's coordinates computed from individual readings and the orientation information. The orientation is computed by forming lines from the sonar readings and then matching these lines with the previously known world model. The final position of the robot is estimated using a maximum likelihood estimate of both

dead reckoning and sensor perception.

In the following sections, we will first give a probabilistic modeling of the uncertainty of the sensors. Then, we will describe the consistency tests we use to eliminate inconsistent data. In section 3.4 we will introduce the perception systems and the process by which the perceived data are converted to location information. Finally in section 3.6 we will discuss sensory data fusion.

3.2 Sensor Uncertainty Modeling

The uncertainty in a particular sensor reading, which is the distance of the actual reflection point to the reading, is modeled using a Gaussian density function [Silk, 1983]. The distribution of the error is also assumed to be Gaussian as the error term is *the* uncertainty in a sensor reading.

It is quite unlikely that one may exactly obtain a probabilistic description of the system. The exact probability distribution is dependent on many physical parameters. Even if it may be approximated after substantial work, the resulting model may be undesirable because of its complexity. We henceforth assume normal distribution for the uncertainty of the sensor readings.

The uncertainty in a reading is the error term related to the reading. The closest approximation we can make about the error term in a sensor reading is the distance of the sensor reading to the estimated reflection point. If the estimated reflection point of the n^{th} reading of the i^{th} sensor, $S_{in} = \left[s_{in_x}, s_{in_y}\right]$, is found to be $R_{in} = \left[r_{in_x}, r_{in_y}\right]$,then the estimate of the error in this reading becomes

$$\varepsilon^*_{in} = mdist(R_{in}, S_{in}) \qquad (3.1)$$

where *mdist()* is a two dimensional, first order Minkowski metric

[Duda and Hart, 1973] which can be stated in our case as,

$$mdist(R_{in}, S_{in}) = \left[|r_{in_x} - s_{in_x}|, |r_{in_y} - s_{in_y}|\right] \tag{3.2}$$

Since the error in the readings is dependent on the environmental constraints, the mean and the variance of the sensor uncertainty function are varying in time. In such cases a maximum likelihood estimate of the mean, which is the arithmetic average of the samples, is used. Geometrically, if we think of n samples as a cloud of points, we would expect the mean of these samples to be the centroid of this cloud. This centroid, referred to as the *sample mean*, is described as the n^{th} arithmetic average of the samples (which are the error terms in our case). The sample mean of the i^{th} sensor is defined as

$$m_{in} = \frac{1}{n} \sum_{j=1}^{j=n} \varepsilon^*_{ij} \tag{3.3}$$

which is the maximum likelihood estimate of the mean for a normal distribution. The sample mean is unbiased, however, we want the model to be sensitive to the recent properties of the environment. Thus, we use the following equation for computing the sample mean

$$m_{in} = \frac{1}{k} \sum_{j=n-k}^{j=n} \varepsilon^*_{ij} \tag{3.4}$$

which biases the sample mean to the last k readings.

As the readings from a single sensor, S_{ij} $j = 1..n$, result from the reflections of many different objects, a means of normalization is required. The normalization should take into account the measure of proximity. This normalization is done as follows

$$\varepsilon_{in} = \varepsilon^*_{in} - m_{in} \tag{3.5}$$

where the asterisk denotes raw or 'unnormalized' data. This method of normalization makes feature values invariant to rigid displacements of the coordinates by forcing the origin of the coordinate system to coincide with the sample mean. The effects of this normalization procedure are shown in Figure 3.1.

The (sample) covariance matrix is:

$$\Lambda_{in} = \frac{1}{k} \sum_{j=n-k}^{n} \left[\varepsilon_{ij}^{*} - m_{in}\right] \left[\varepsilon_{ij}^{*} - m_{in}\right]^{T} \tag{3.6}$$

The probability of ε_{in}, the n^{th} estimated error of sensor i from our assumption of normal distribution becomes:

$$f(\varepsilon_{in}^{*}) = \frac{1}{2\pi \left|\Lambda_{in}\right|^{\frac{1}{2}}} exp\left[\frac{-1}{2} \left[\varepsilon_{in}^{*} - m_{in}\right]^{T} \Lambda_{in}^{-1} \left[\varepsilon_{in}^{*} - m_{in}\right]\right] \tag{3.7}$$

which will be used as the weight related to the n^{th} reading of the i^{th} sensor. This equation is based on the normal distribution characteristics of the sensors' data.

After the localization process is complete the last set of the sensor readings' reflection point estimates are updated using a linear approximation of the final estimated position. This is shown in Figure 3.2.

3.3 Consistency Test of a Single Sonar Reading

The sensor data, generated by the reflection of the sonar waves coming back to the transducer, does not always give reliable information about the positions of the objects in front of the sonar sensor. This is either caused by multiple reflections as shown in Figure 3.3, or by the fact that the angle between the normal to the surface of the reflection and the transducer beam is greater than the transducer's half angle, thus we must distinguish consistent reflections from inconsistent ones. For this purpose a number of consistency checks are applied to the sensor points. We will explain these in the following paragraphs.

The consistency checking on the sensor data is made in two steps. First a set of empirical heuristics are applied, and then a probabilistic consistency evaluation based on a Gaussian model is computed.

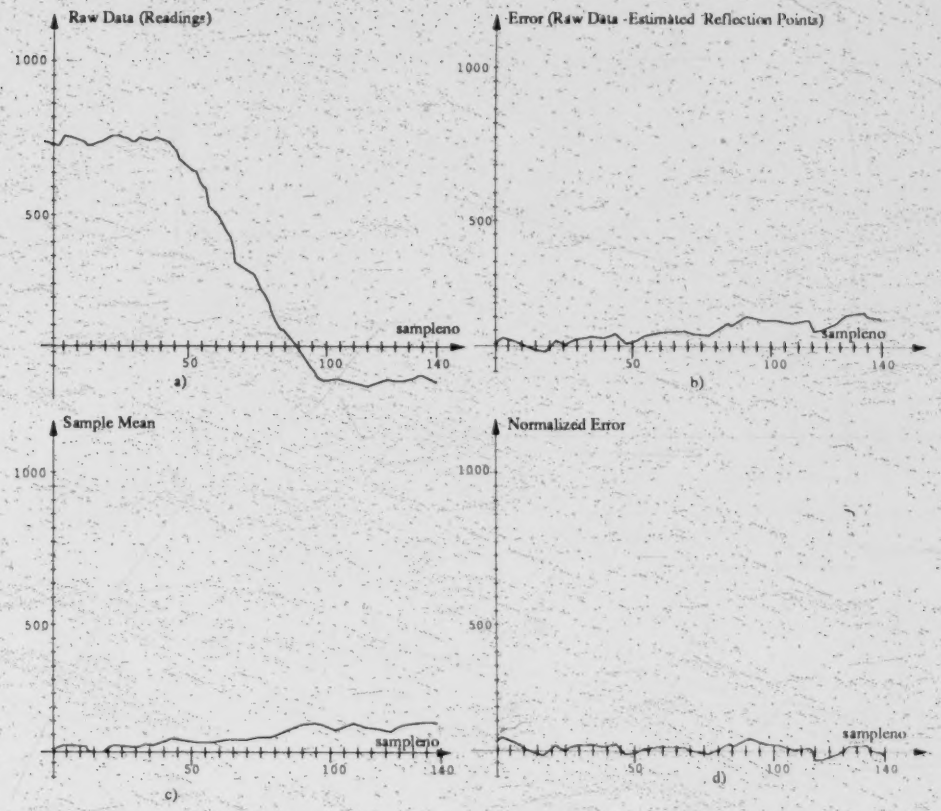

Figure 3.1: Normalization of sensor data

By this set of tests we first try to remove inconsistent data which we call *false reflections*. Examples of false reflections may be the points that are caused by multiple reflections or those with an error that the system should not tolerate (Figure 3.3). The following rules are used in the decision process.

Visibility Test One constraint that the world model places on any possible reflection is that the reflecting point must be directly visible by the transducer (Figure 3.5). The existence of an object between a candidate reflection point and the sensor implies the penetration of sonar rays through the object and consequently the rejection of the candidate.

Proximity Test A candidate reflection point must lie within a certain proximity

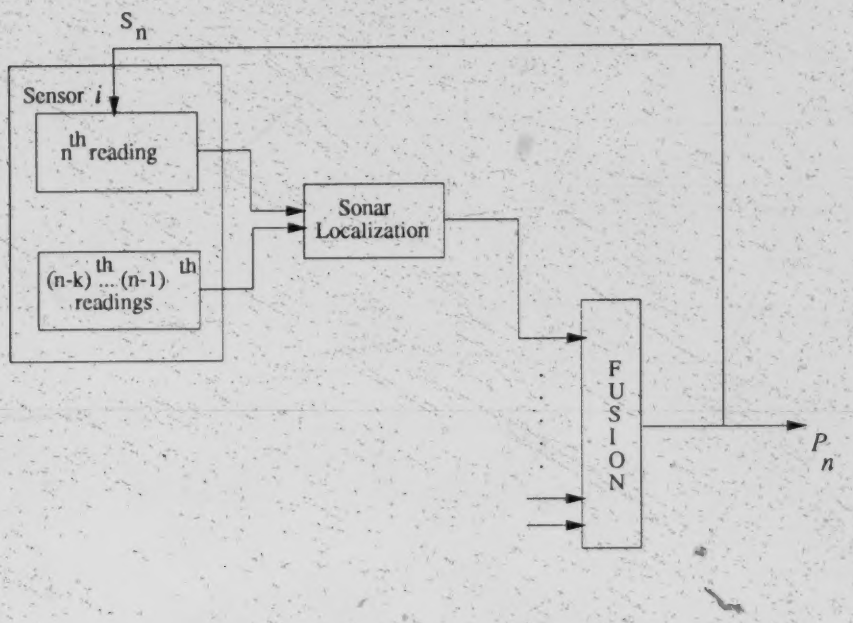

Figure 3.2: Sensor feedback

threshold from the reflecting surface (Figure 3.6). The threshold value for the proximity test is set empirically.

Cone of Reflection Test Any candidate of a reflection point must be within a certain neighborhood of the cone of reflection (Figure 3.4), i.e., the angle between the sonar contour and the normal to the surface must be within a certain limit as shown in Figure 3.4.

Selection Among Multiple Candidates If multiple matchings are made between the reflection points and the set of candidate reflectors of the world model, then we choose the candidate with the shortest distance to the reading.

After running all these tests on a set of scan, each consistent point is related to a point of a line segment. The other points, which are eliminated through these tests,

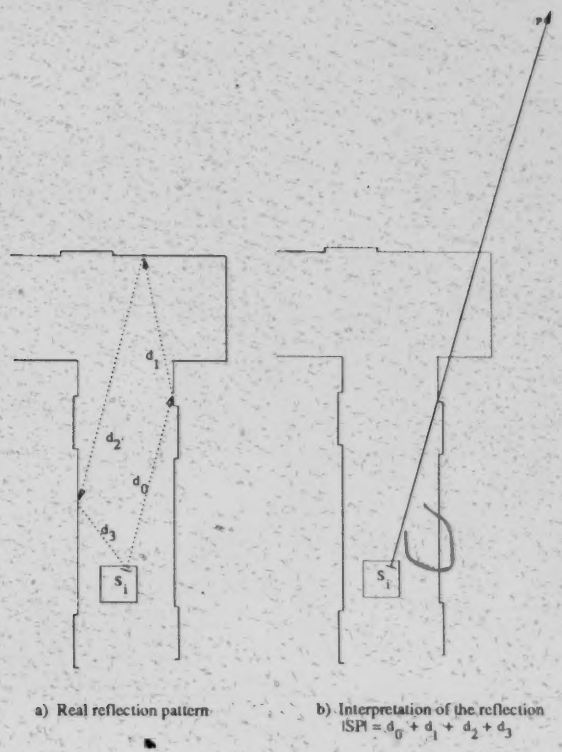

Figure 3.3: A multiple reflection pattern and its interpretation

are not used at all. Multiple reflection patterns can be searched for, and calculated for the eliminated points, but for our purposes, the time complexity of the process is not feasible. There may also be multiple (even infinite) solutions for such patterns. Thus, multiple reflection points are considered useless.

Finally, the weights are assigned to the points which pass the tests using equation 3.7. These weights define the system's degree of belief in the points, and are used later for line extraction.

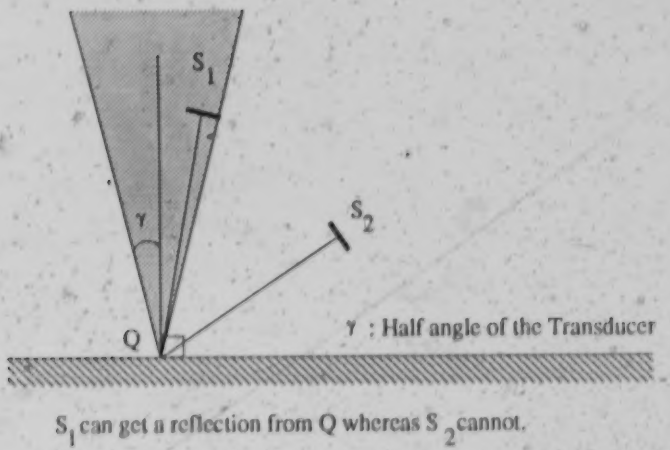

Figure 3.4: Cone of reflection

3.4 Perception Models

3.4.1 Monitoring the Motion by Dead-Reckoning

As the robot proceeds, the movements are simultaneously monitored using the encoders attached to the wheels. Readings from the encoders are mapped to the world coordinates, to determine the change in the position of the robot. Then this information is integrated to give the robot position vector $\vec{P}$ by Equation 3.8, where the encoder readings are $E = (e_r, e_l)$, the distances traveled by the right and left wheels respectively.

$$\vec{P} = \int_{t_0}^{t_1} \mathbf{K}\left(\frac{\delta E}{\delta t} dt\right) \tag{3.8}$$

In the above equation $\mathbf{K}$ is the transformation which will be explained in section 4.3, mapping the differential changes in the robot's wheel positions into the world coordinates. From 3.8, differential changes are given by:

$$d\vec{P} = \mathbf{K}\left(\frac{\delta E}{\delta t} dt\right) \tag{3.9}$$

When the changes in the encoders are finite, the finite change in $\vec{P}$ is approximated

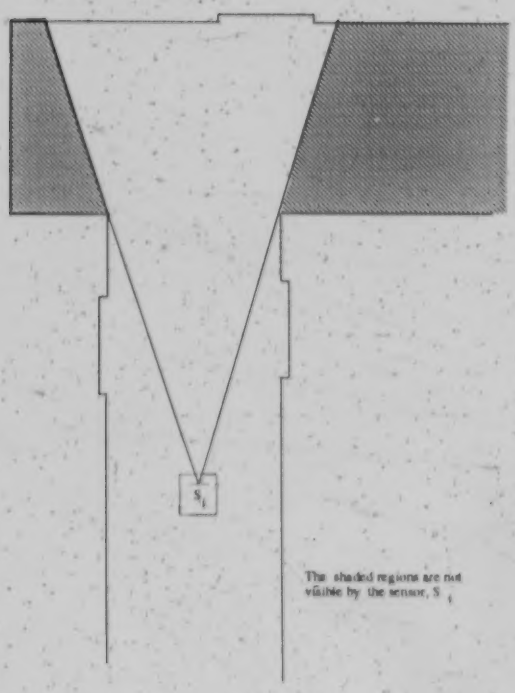

Figure 3.5: Visible space

by:

$$\Delta \vec{P}_i = \mathbf{K} \Delta E_i \tag{3.10}$$

where

$$\Delta E_i = E_i - E_{i-1}, \tag{3.11}$$

Numerically, equation 3.8 is approximated by,

$$\vec{P}_i = \vec{P}_{i-1} + \Delta \vec{P}_i \tag{3.12}$$

Equation 3.12 is used along with the encoder readings for the continuous monitoring of the robot's position.

3.4.2 Monitoring the Motion by Sonar Sensors

The Cartesian coordinates of the base coordinate frame of the robot, with an

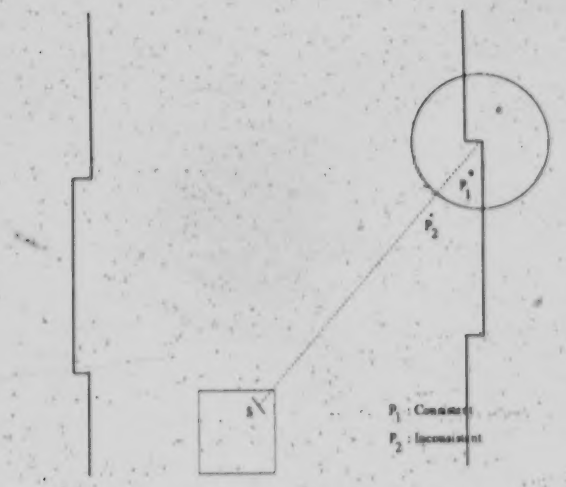

Figure 3.6: Nearness Test

orientation of θ, are computed for each sonar reading by the following equations derived from Figure 3.7

$$X_R = x_P - (l_{S_i} + d_{S_i})(S(\gamma_i + \theta)) \tag{3.13}$$

$$Y_R = y_P - (l_{S_i} + d_{S_i})(C(\gamma_i + \theta)) \tag{3.14}$$

where (X_R, Y_R) is the coordinates of the robot, (x_P, y_P) is the reflection point, l_{S_i} is the distance between the center of the robot and the sonar sensor, d_{S_i} is the distance measured by the sonar sensor, and γ is the angle that the sensor beam makes with the robot's y axis. The final approximation of these points are made using the equations

$$\overline{x} = \frac{\sum_{i=1}^{N} w_{x_i} x_i}{\sum_{i=1}^{N} w_{x_i}} \tag{3.15}$$

$$\overline{y} = \frac{\sum_{i=1}^{N} w_{y_i} y_i}{\sum_{i=1}^{N} w_{y_i}} \tag{3.16}$$

The above equations give the weighted averages of the sonar readings, where w_{x_i}, w_{y_i} are the computed weights for each reading x_i, y_i of the reflection point p_i.

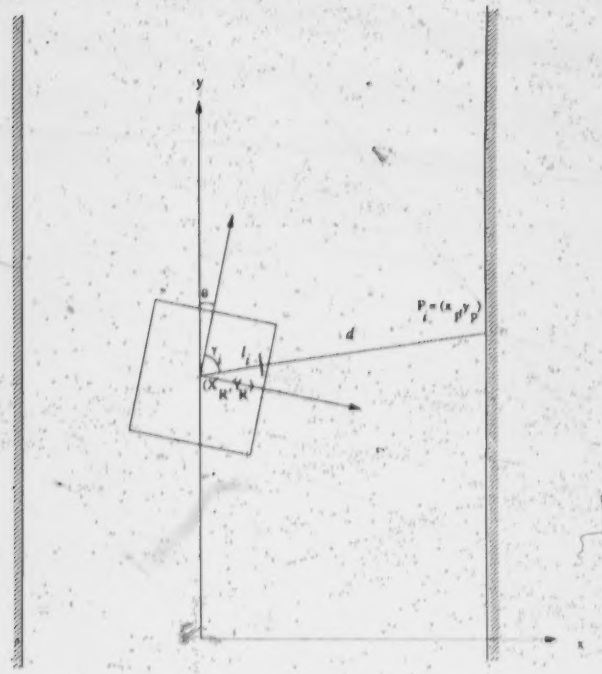

Figure 3.7: Geometry of a reflection

3.5 Orientation Determination

In addition to position localization, the orientation of the robot must also be localized. Orientation is of extreme importance as the trajectory generation depends on the orientation. Orientation cannot be calculated from a single point because a single point does not contain any orientation information. Instead we obtain it by forming lines from multiple sensor readings and matching these lines with the a priori world model.

The points are related to a line segment in the world model, as mentioned in the previous section. Thus, the clustering process of the points is already done. These points are then used to form a line using a weighted least square fit. The information about the world model is integrated into the system by assigning a weight to each point. This weight is the uncertainty in the reading.

The line forming algorithm is mainly composed of two steps:

I. Group all points which are related to multiple segments of a single wall.

II. Form a line using weighted least square fit method.

The basic formulation of the weighted least square fit is as follows:
A single scan of N sensors yields N measurements. The linear equations from these measurements form two N-dimensional vectors, X and Y, which are of the form

$$Y = [X\ \mathbf{1}] \begin{bmatrix} a \\ b \end{bmatrix} \tag{3.17}$$

where $[X\ \mathbf{1}] = \begin{bmatrix} x_1 \ldots x_N \\ 1 \ldots 1 \end{bmatrix}^T$ and $Y = [y_1 \ldots y_N]^T$ are the coordinates of the N readings.

The weighted least square approximations of parameters a and b are given by [Thomas, 1987]

$$\begin{bmatrix} a \\ b \end{bmatrix}_{LS} = \left[[X\ \mathbf{1}]^T R [X\ \mathbf{1}] \right]^{-1} [X\ \mathbf{1}]^T RY \tag{3.18}$$

R is the $N \times N$ diagonal weighting matrix where the diagonal elements are the computed weights of the sensors' readings. Figure 3.8 shows a sample fit. The lines resulting from these points are compared with their corresponding segments in the world model to find the estimated orientation error. This error is then fused into the system as we will explain in the next section.

3.6 Final Fusion of the Data

The Maximum Likelihood method is used in the fusion of the data coming from the two types of sensors. We apply the method using the conditional probabilities of the sensor observations, $\mathcal{S}$, given that the estimated position is $\mathcal{P}$.

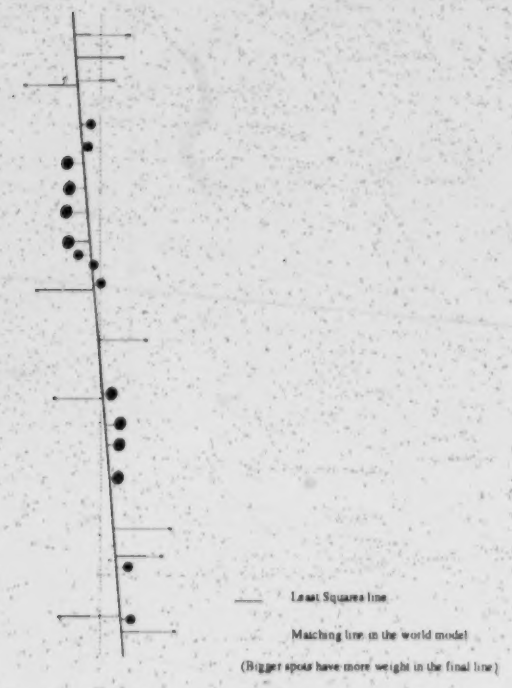

Figure 3.8: A sample weighted least square fit

In the decision process, the maximum likelihood estimator is obtained by maximizing the likelihood defined by the conditional density function

$$l = p(\mathcal{P} \mid \mathcal{S}) = \frac{p(\mathcal{S} \mid \mathcal{P})p(\mathcal{P})}{p(\mathcal{S})} \tag{3.19}$$

with respect to $\mathcal{S}$. In other words, we compute $\mathcal{P}$ such that the following equation is maximized

$$l = p(\mathcal{P} \mid \mathcal{S}) = \prod_{i=1}^{n} p(\mathcal{P} \mid \mathcal{S}_i) \tag{3.20}$$

where, n is the number of sensor outputs and $\mathcal{S}_i$ is the output of i^{th} sensor.
Equation 3.20 can be maximized by maximizing its natural logarithm.

$$L(\mathcal{P}) = \sum_{i=1}^{n} ln(p(\mathcal{P} \mid \mathcal{S}_i)). \tag{3.21}$$

When we use our assumption that the sensors are modeled by a normal distribution function we get

$$L(\mathcal{P}) = \sum_{i=1}^{n} ln\left(\frac{1}{(2\pi)^{\frac{n}{2}} \mid \Lambda_i \mid^{\frac{1}{2}}} exp\left(\frac{-1}{2}[\mathcal{S}_i - \mathcal{P}]^T \Lambda_i^{-1} [\mathcal{S}_i - \mathcal{P}]\right)\right) \tag{3.22}$$

Figure 3.9: Detecting a deviation from the estimated path

where Λ_i is the $n \times n$ covariance matrix.

We can obtain the maximum likelihood estimate for $\mathcal{P}_{MLE}$ by maximizing equation 3.22. Differentiating it with respect to $\mathcal{P}$ and setting it to zero, we get

$$\frac{\delta L(\mathcal{P})}{\delta \mathcal{P}} \Big|_{\mathcal{P}=\mathcal{P}_{MLE}} = 0 \tag{3.23}$$

Solving the above equation using equation 3.22 we get [Sage and Melsa, 1971]

$$\mathcal{P}_{MLE} = \left[\sum_{i=1}^{n} \Lambda_i^{-1}\right]^{-1} \left[\sum_{i=1}^{n} \Lambda_i^{-1} \mathcal{S}_i\right] \tag{3.24}$$

In our case where n = 2 the equation becomes

$$\mathcal{P}_{MLE} = \left[\Lambda_1^{-1} + \Lambda_2^{-1}\right]^{-1} \left[\Lambda_1^{-1}\mathcal{S}_1 + \Lambda_2^{-1}\mathcal{S}_2\right] \tag{3.25}$$

which also states that the weights of each sensor are inversely proportional to their variances.

The computation of the covariances from the sensor readings is not always an easy task because of the inadequate number of available samples. Thus, the question of how to proceed arises. A solution is to reduce the dimensionality by redesigning the feature extractor [Duda and Hart, 1973]. The feature extractor in our case

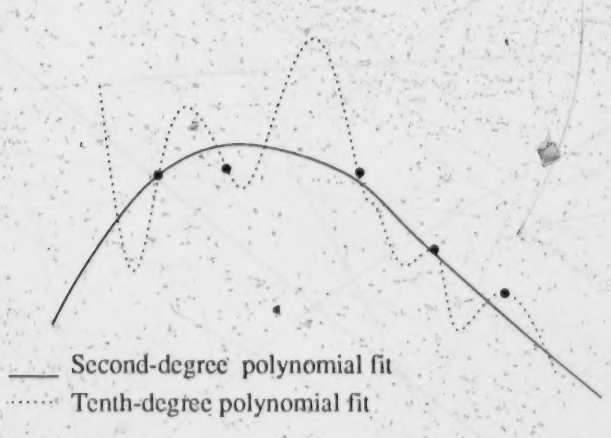

Figure 3.10: Fitting curves to a set of data points

is the position of the robot composed of three identifiers, the two coordinates and the orientation of the robot,(x, y, θ). Reduction can be done by computing only one or two of these identifiers. To provide stability of the solution, the correlations can be heuristically removed by thresholding the sample covariance matrix[1]. It may be assumed that all covariance for which the magnitude of the correlation coefficient is smaller than a value is actually zero. We use both of the two methods for our application. This solution is obviously suboptimal, but under the existence of the problem of insufficient data, the resulting heuristic estimates often constitute an acceptable solution. The reasons for these are explained briefly in [Sage and Melsa, 1971]. In an estimation process a large amount of samples would be needed to get a good fit for higher order solutions to the problem. For example a tenth degree polynomial can fit perfectly to a set of sample points where one can hardly expect such a curve to fit new data well; therefore, a second degree polynomial solution to the set of data is preferred. Figure 3.10 simply illustrates this problem. Although the curve formed by the tenth degree polynomial fit exactly passes through the data points, it is usually unlikely that the next data points will be consistent with this curve.

[1] As the data is normalized, the biggest effect of this method may be assigning an equal weight to each sensor.

3.7 Summary

We have introduced a method for sensor data interpretation in order to provide a means of localization for the robot. This method helps the robot to position itself using multiple sensors. The basic aspect of the method is that it employs the current estimated position of the robot in an iterative algorithm to provide localization.

A probabilistic modeling of the sensors is used to obtain the weight of sensor readings in position determination. The physical considerations are done for sonar sensors and differential encoders. However, the method is extendable to other sensory systems. The required modifications will be introducing the localization method for the sensor. Finally, a general data fusion scheme is used to achieve a final position estimate. The fusion process does not make any particular assumptions about the types of sensors, except that the sensor uncertainties are normally distributed.

The method is implemented in our mobile robot system. The computations are time efficient, so the localization process, along with the route planning which will be explained in Chapter 4, can be done efficiently in real-time. Some test results are presented in Chapter 5

Chapter 4

Trajectory Planning and Execution

4.1 Introduction

Trajectory planning has always been an active research area in robotics. Many trajectory generation methods have been described in literature for both robot manipulators and mobile robots. For the case of a mobile robot, navigating in an environment which is not strictly constrained (e.g. within corridors), a complex algorithm will be very time consuming. Thus, straightforward path generation methods are preferred. Also, because of servo errors, path planning for robot motion must be frequently repeated. In some autonomous vehicle controlling systems employing sensor feedback [Kanayama and Miyake, 1986] [Wallace *et al.*, 1986], a central control program updates the trajectory to be followed every T-seconds, where T is the processing time of the sensory module. The navigation of a mobile robot is similar to that in Figure 4.1. The control program receives a sequence of data related to the current position and orientation of the vehicle, and generates trajectories in real time.

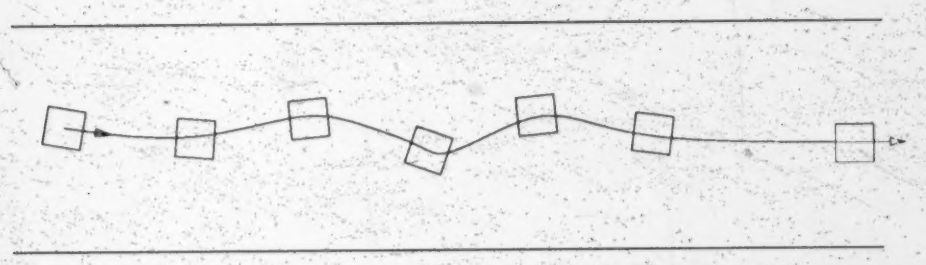

Figure 4.1: A sample trajectory for the mobile robot

Current studies involve many levels of computation in trajectory generation. One of the main approaches is solving the problem using polygonal paths, and smoothing the intersections of the straight line segments with linear curvatured geometries to maintain continuity with respect to the physical movement capabilities of the robot at hand [Kanayama and Miyake, 1986] [Tsumura *et al.*, 1981]. The constraints restricting the movement of the robot become of utmost importance when the environment is complex. In such cases, trajectory generation becomes a complicated task, involving the detailed investigation of the geometric aspects of the robot and the environment [Laumond, 1989] [Tournossoud, 1988]. Even though these trajectories are quite detailed and hard to control, they are theoretically safe.

In the next few sections, after a brief review of the existing methods in mobile robot motion planning, we will describe the kinematics of the robot. We will also explain a new method we use to generate trajectories, based on previous ones found in the literature. Then we will concentrate on the execution of the trajectory we generated and the motion control of the robot, which, in our experiment is the TRC Labmate [Tra, 1989b].

4.2 Motion Planning of Mobile Robots

Trajectory planning is only one aspect of the navigation problem. The planning process involves issues such as environment modeling and perception, ac-

counting for inaccuracies, real time decision making, and special structure learning [Chatila and Giralt, 1986]. One can classify the algorithms for trajectory planning into the following three categories: local methods, global methods and highly constrained methods. The classification is done in terms of the a priori knowledge that the robot has about the environment and the constraints the environment imposes on the robot.

1. **Local Methods** These methods assume no knowledge about the environment and generate trajectories by making use of local, that is usually poor, but quickly acquired information about the environment, in real time. The approach of using potential fields is a good example [Khatib, 1986]. The robot assumes itself to be going in a fictitious potential field wherein obstacles are associated with repulsive fields, and the goal is an attractive field. The obstacles can be thought of as hills and the robot, modeled as a ball, rolls towards the goal. The potential field method is quite useful in situations involving convex obstacle avoidance, but it has many drawbacks in constrained spaces, where the obstacles are very near. [Faverjon and Tournossoud, 1987] state some of the drawbacks of this method by approaching the problem wherein collision free constraints appear as linear constraints in a quadratic criterion minimization problem associated to the goal. As it uses only a local view of the environment, these methods do not guarantee that an existing solution will be found.

2. **Global Methods** These methods deal with the cases where a global map of the environment is known, so the trajectory planning is done globally. There are several methods dealing with these cases. Particular approximation schemes are used to structure the Euclidean space. One method assumes the robot to be circular [Laumond, 1987] and uses a generalized visibility graph in a general environment. Trajectories thus produced are smooth without sharp angles. Some methods decompose the environment into elementary spaces [Brooks and Lozano-Perez, 1983] which are structured into a graph whose adja-

cency relation indicates the possibility of moving from one place to another.All these methods are applicable to cases where the free space is large with respect to the robot's geometric and kinematic constraints.

When the environment is highly constrained, there is a need for the formalization of the configuration space **CS**. In [Laumond, 1987] the space is formed of independent parameters that characterize the position and orientation of a mobile body. **CS** is divided into many subsets (subspaces). These are: the admissible space, ACS, where the robot does not intersect with obstacles, the free space, FCS, which is the closure of the interior, and the occupied space, OCS, which is the complement of ACS. The trajectory is generated by searching into the **CS** and comparing the feasibility of the formed trajectories.

3. **Highly constrained methods** The last class of methods deals with complex environments where there are a several constraints restricting the movement of the robot. These methods are generally referred to as the piano mover problem. [Schwartz and Scharir., 1983] state the problem as representing the free space in the configuration space as a connectivity graph of free regions, which can be searched for a path. The free space is limited for the robot, and the trajectories generated under geometric constraints employ a number of maneuvers to change the direction of travel. Thus the execution of these methods is quite complex.

In our study, we are concerned with both the global and the local methods of trajectory generation. Moreover, we will concentrate on the execution of the trajectory. The free space is regular, i.e. there exists a finite set of straightforward transformations which can carry the robot between any two points in the free space [Tournossoud, 1988]. The time spent in real time sensor interpretation and trajectory generation is important, so a simple and robust real-time algorithm is required.

4.3 Kinematics of the TRC Robot

In this section we will consider some basic kinematic relations describing the TRC Robot's motion synthesis. They include the relationship between external world coordinates and angular or linear displacements of the robot, the Jacobian matrix that relates the robot's wheel velocities to its velocity in Cartesian coordinates, and another Jacobian matrix that relates velocities in the robot's Cartesian coordinate frame to the world coordinate frame.

The TRC mobile robot is a vehicle with two parallel wheels mounted on the same axis and controlled independently. There are four free wheels on the corners of the robot. As mentioned before, we define its position and orientation by a configuration vector which is a 3-tuple (x, y, θ), where x and y are the Cartesian coordinates of the origin of the robot and θ is its orientation.

Another property of the TRC Robot is that it is non-holonomic[1]. The differential equation

$$dy = tan(\theta)dx,$$

shows that any of the configuration parameters can be written using the other two. It also states that the robot (actually, the wheels) can only move in the direction normal to the axis of the driving wheels.

In order to analyze the kinematics of the robot, first of all we must derive the equations of motion. Using Figure 4.2, the following are derived for the TRC robot.

$$\dot{x} = -S\theta\frac{v_r + v_l}{2} \tag{4.1}$$

$$\dot{y} = C\theta\frac{v_r + v_l}{2} \tag{4.2}$$

$$\dot{\theta} = \frac{v_r - v_l}{2d} \tag{4.3}$$

[1]The number of configuration parameters is less than the degrees of freedom. In our case the configuration is defined by three parameters, whereas the degree of freedom is two.

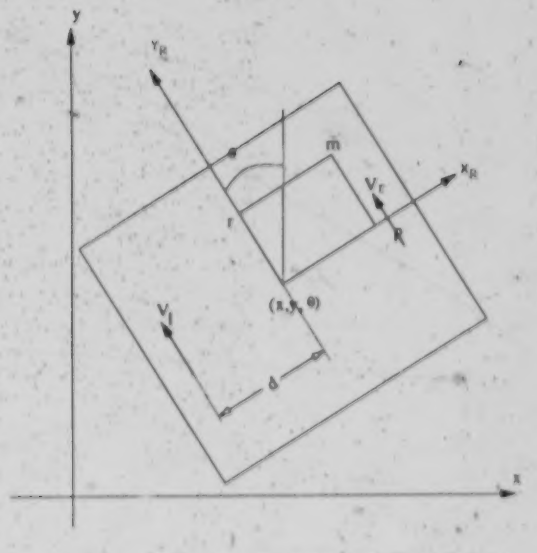

Figure 4.2: The TRC in World Coordinates

where, d is the lateral distance between the Y axis of the robot and the wheels. v_r and v_l are the velocities of the right and left wheels respectively. Let us define

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{4.4}$$

The only control variables we have are the velocities of the two wheels, and they are related to $\dot{q}$ by $\dot{q} = \mathbf{K}(\theta)v$, where

$$\mathbf{K}(\theta) = \frac{1}{2}\begin{bmatrix} -S\theta & -S\theta \\ C\theta & C\theta \\ \frac{1}{d} & -\frac{1}{d} \end{bmatrix} \tag{4.5}$$

$$v = \begin{bmatrix} v_r \\ v_l \end{bmatrix} \tag{4.6}$$

The matrix $\mathbf{K}$ is useful for mapping v into $\dot{q}$, but inverse kinematic analyses cannot be made as $\mathbf{K}^{-1}$ cannot be computed to do the inverse mapping. In order to

find a non-singular matrix which relates the robot's coordinates to the world coordinates, we will introduce the non-holonomicity constraint into the equation, and derive a Jacobian matrix which is a function of θ. For any point m, its coordinates (p, r), defined with respect to the origin of the robot, and its coordinates (m_x, m_y), defined with respect to the world coordinate frame, are related by: [Tournossoud, 1988]

$$m_x = x - rS\theta + pC\theta \quad (4.7)$$

$$m_y = y + rC\theta + pS\theta \quad (4.8)$$

$$\dot{m} = \frac{dm}{dt} \quad (4.9)$$

$$\dot{m}_x = \dot{x} - (rC\theta)\dot{\theta} - (pS\theta)\dot{\theta} \quad (4.10)$$

$$\dot{m}_y = \dot{y} - (rS\theta)\dot{\theta} + (pC\theta)\dot{\theta} \quad (4.11)$$

where x, y, and θ are the coordinates of the center of the robot. From 4.4, 4.10,and 4.11, we get

$$\dot{m} = \mathbf{L}(\theta)\dot{q} = \begin{bmatrix} 1 & 0 & (-rC\theta - pS\theta) \\ 0 & 1 & (-rS\theta + pC\theta) \end{bmatrix} \dot{q} \quad (4.12)$$

The standard differential relation $\dot{m} = \mathbf{J}v$, where $\mathbf{J}$ is the Jacobian matrix at point m for configuration q is derived from 4.12 and 4.4 as follows [Tournossoud, 1988]

$$\dot{m} = \mathbf{L}(\theta)\dot{q} = \mathbf{L}(\theta)\mathbf{K}(\theta)v \quad (4.13)$$

$$\mathbf{J}(\theta) = \mathbf{L}(\theta)\mathbf{K}(\theta) \quad (4.14)$$

$$\mathbf{J}(\theta) = \frac{1}{2}\begin{bmatrix} (C\theta - \frac{1}{d}(rC\theta + pS\theta) & (S\theta + \frac{1}{d}(rC\theta + pS\theta) \\ (C\theta - \frac{1}{d}(rS\theta - pC\theta) & (S\theta + \frac{1}{d}(rS\theta - pC\theta) \end{bmatrix} \quad (4.15)$$

The Jacobian matrix above is used for the motion control of a trajectory, using the inverse kinematic equation $v = \mathbf{J}^{-1}(\theta)\dot{m}$. Also the constraint checking of any point on the robot, to ensure that a generated trajectory is safe, is done by Equation 4.15.

The speeds of the wheels are set to achieve the desired curvature, which is computed by the trajectory planning module. The radius of curvature, ρ, of a trajectory

(see Figure 4.3) is calculated by:

$$\rho = d\left(\frac{v_r + v_l}{v_r - v_l}\right) \tag{4.16}$$

from the above equation we derive:

$$\frac{v_r}{v_l} = \frac{\rho + d}{\rho - d} \tag{4.17}$$

Thus, the ratio of the speeds of the two wheels can be set to achieve any desired curvature.

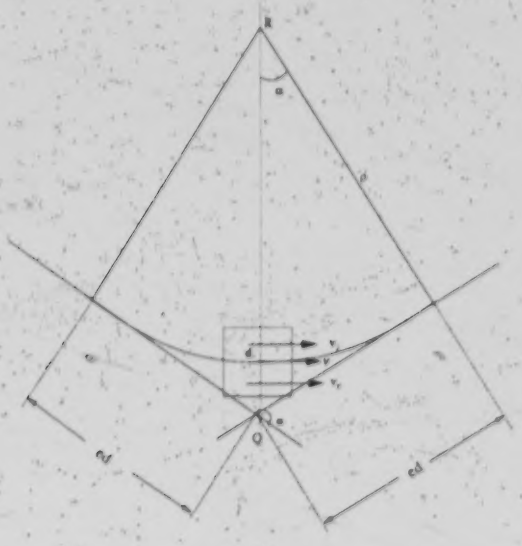

Figure 4.3: The Turn Motion

Let $s(t)$ be the distance traveled within a certain time t. It is computed using the following equation:

$$s(t) = \int \sqrt{\dot{x}^2 + \dot{y}^2}dt \tag{4.18}$$

The equations given above are used both for monitoring the motion of the robot and for the execution of a generated trajectory.

4.4 Trajectory Generation

The purpose of this section is to give a method of describing trajectories for mobile robots. The basic property that the motion generation algorithms should possess is the provision for the functionality of the motion, i.e. to provide continuity and smoothness of motion. Besides this basic requirement, additional properties are also needed to increase the adaptivity and feasibility of the robot. For example the algorithm should provide the proper adaptivity for multiple sensors, i.e. it should be accepting modifications because of possible position errors detected by the proximity sensors.

We will first give a formal definition of the problem of finding a trajectory, and then we offer a solution which divides the main goal of reaching a final configuration into a finite set of sub-goals. A trajectory will be generated for each of these sub goals, where the continuity of the complete trajectory is guaranteed. At the end of the section we will examine the possible problems we may encounter.

4.4.1 Basic Formulation of the Solution

We will approach the problem by dividing it into sub-goals and then finding a solution to these sub-goals, which can be added together to form a solution to the original problem.

We intend to use a method that generates the trajectory with a finite sequence of distinct curves, and specify each curve with as little information as possible[2].

We define the solution for a trajectory as follows:

[2]Since the trajectory information will be sent from the controller to the vehicle, the amount of control data should be as little as possible.

Definition 4.1 *Let* $\mathbf{P}$ *be a sequence of position vectors that define the trajectory the robot is to follow, where*

$$\mathbf{P} = (P_0, P_1, ..., P_n) = ((x_0, y_0, \theta_0), ..., (x_n, y_n, \theta_n))\ n > 0$$

If a directed curve $\mathbf{C}$ *starts at* P_0*, ends at* P_n *and passes through each* P_i *in order, then* $\mathbf{C}$ *is said to be a* `solution` *of* $\mathbf{P}$*. We denote the solution by the relation :* $\mathcal{S}(\mathbf{C}, \mathbf{P})$.

Let C_i be a trajectory formed of a finite number of segments with constant curvatures. From the above definition we may conclude that, if C_i satisfies the position transformation P_i $(i = 0, .., n)$, $\mathcal{S}(C_i, P_i)$, then the sequence of directed curves $\mathbf{C} = (C_0, ..., C_n)$ satisfies the transformation $\mathbf{P} = (P_0, ..., P_n)$ which is the trajectory. In other words:

$$(\mathcal{S}(C_i, P_i), i = 0, .., n) \Rightarrow (\exists \mathbf{C} \mid \mathcal{S}(\mathbf{C} = (C_0, ..., C_n), \mathbf{P} = (P_0, ..., P_n)))$$

The aforementioned sub-goals are found by connecting the initial and final configuration using straight lines. These lines must be within the free space, **FS** (see Figure 4.4). The intersection points of these lines, which are called the turn points, will be connected using a curvature to provide continuity of the motion. [Kanayama and Miyake, 1986] uses a clothoid to approximate a continuous curvature in solving the problem. However, for our purposes a circular arc is more appropriate as the control is simpler.

Let a segment be a continuous path function between two points where the curvature remains constant throughout the segment. Each trajectory is defined by a sequence of segments, and each segment can either be a straight line or an arc. Thus, each segment can be expressed by a two tuple (ρ, α), which are the radius of curvature and the amount of change in the orientation respectively (Figure 4.5). Any trajectory is then represented by a sequence $\tau = ((\rho_0, \alpha_0), ..., (\rho_n, \alpha_n))\ n \geq 0$. This is the output of the path generation function.

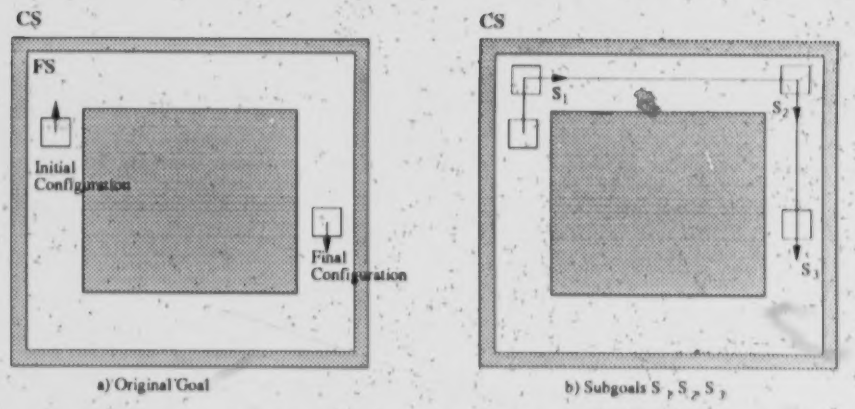

Figure 4.4: Dividing the goal into sub-goals

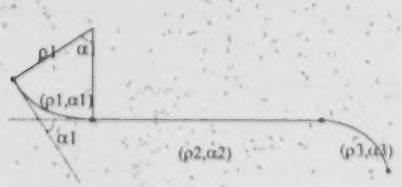

Figure 4.5: Segments of a trajectory

Solutions to the problem can be classified according to the number of non-linear segments they include by calling the number of curves in the solution as the *degree* of the solution. Our algorithm tries to find solutions of at most degree two. The following are the rules used in the algorithm for generating a trajectory for two position vectors P_0 and P_1.

In [Kanayama and Miyake, 1986] the authors offer a trajectory as in Figure 4.6a. Since such a motion is not desired within a corridor as it may bring the robot too close to a wall, we adopt a trajectory of the form in Figure 4.6b.

To compute motion parameters for a degree two solution we introduce the *exit distance, ed.*

Definition 4.2 *The exit distance is the distance between the start of the turn and the intersection point Q (See Figure 4.3).*

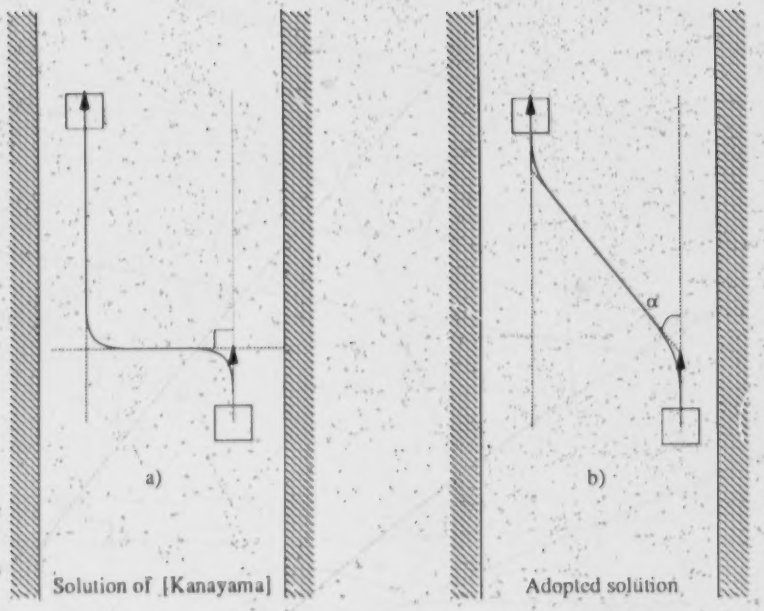

Figure 4.6: Alternative solutions for trajectory generation

Then the trajectory for any sub-goal can be computed as follows: a straight line is drawn between two points (Q_0, Q_1) which are ed away from the initial and final points respectively. Then the radii of curvature and the turning angle can be calculated according to this line and the exit distance (see Figure 4.10). The computation of these parameters are explained at the end of this section.

4.4.2 Determination of the Solution Degree and Exit Distance Selection

Assuming there are no obstacles along the straight line segments between two sub-goals, if an intersection exists between the two lines tangent to the position vector pair at hand (see Figure 4.7a), a solution of degree one exists. However, in some cases, where the intersection point Q does not lie in the free space, this solution is not desirable. For such cases we select a solution of degree two (See Figure 4.7b).

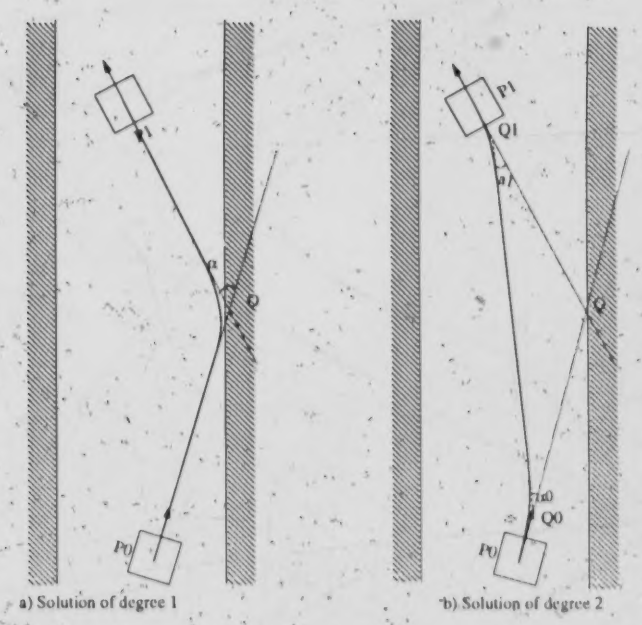

Figure 4.7: Two different solutions for the same trajectory

Note that the radius of curvature is computed with an assigned exit distance. There are two cases where the default exit distance may have to be changed:

1. If the vehicle is likely to collide with an object, as in Figure 4.8a within the default exit distance, then exit distance is tuned accordingly(see Figure 4.8b).

2. If the remainder of the motion segment is less than the default exit distance, as shown in Figure 4.9a, the exit distance must be changed to that shown in Figure 4.9b.

Calculation of the Radius of Curvature

Finding the intersection points of the lines forming the trajectory has been explained in the previous section. Now, we will derive the equations that allow us to compute the radius of curvature, ρ, of the arc which connects pairs of segments. The

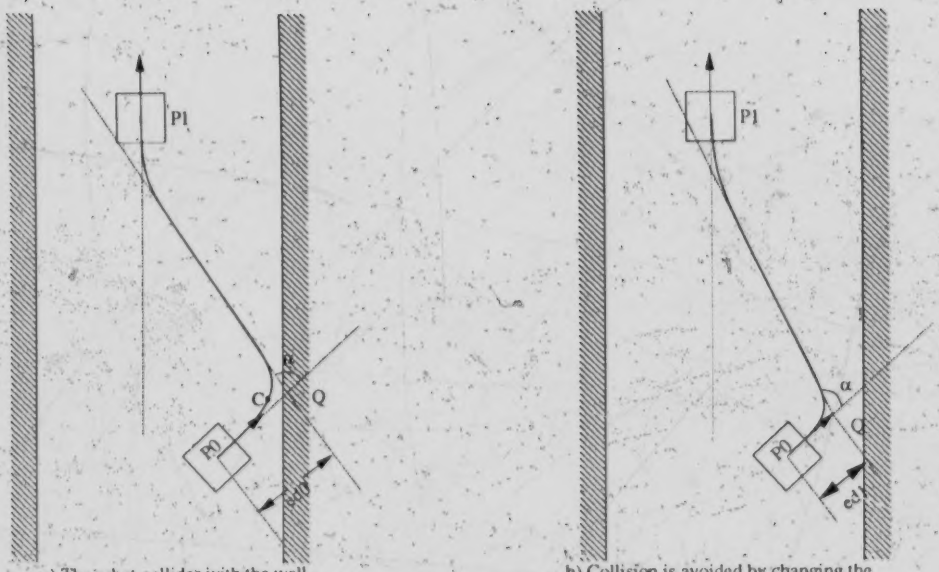

Figure 4.8: Avoiding collision by changing the exit distance

property used in calculation of ρ is that the arc is tangent to the linear paths at a distance equal to the assigned exit distance. The intersection of the normals to the lines at these points will be the center of the arc, and the radius of curvature will be the distance from this point to one of the lines.

The calculation uses the following input parameters: P_0 and P_1, which are the initial and final configurations, and β_0 and β_1 are the initial and final orientations.

The calculation process is as follows (refer to Figure 4.10): first, we find the point Q_0 using the fact that it is at a distance ed away from P_0 in the direction of the initial orientation.

$$\overline{P_0Q_0} = ed$$

$$Q_0 = (q_{0_x}, q_{0_y}) = \left(P_{0_x} + edS\beta_0, P_{0_y} + edC\beta_0\right)$$

Using the same fact, Q_1 becomes,

$$Q_1 = (q_{1_x}, q_{1_y}) = \left(P_{1_x} + edS\beta_1, P_{1_y} - edC\beta_1\right).$$

a) The robot passes the destination within exit distance

b) Sharpness is increased by decreasing the exit distance

Figure 4.9: Destination constraint on the exit distance

In order to find the coordinates of the point S, we will first find the angle γ that the line $\overline{Q_0Q_1}$ makes with the Y axis, that is:

$$\gamma = arctan\left(\frac{(q_{0_x} - q_{1_x})}{(q_{1_y} - q_{0_y})}\right)$$

the turning angle, α, is

$$\alpha = \gamma + \beta_0$$

and S becomes,

$$S = (s_x, s_y) = \left(q_{0_x} - edS\gamma, q_{0_y} + edC\gamma\right)$$

At this point we draw two lines l_0 and $l1$ the normals to the trajectories at points P_0 and S respectively. These normals to the lines $\overline{P_0Q_0}$ and $\overline{Q_0S}$ intersect at the center of the arc.

$$R_0 = intersection(l_0, l_1)$$

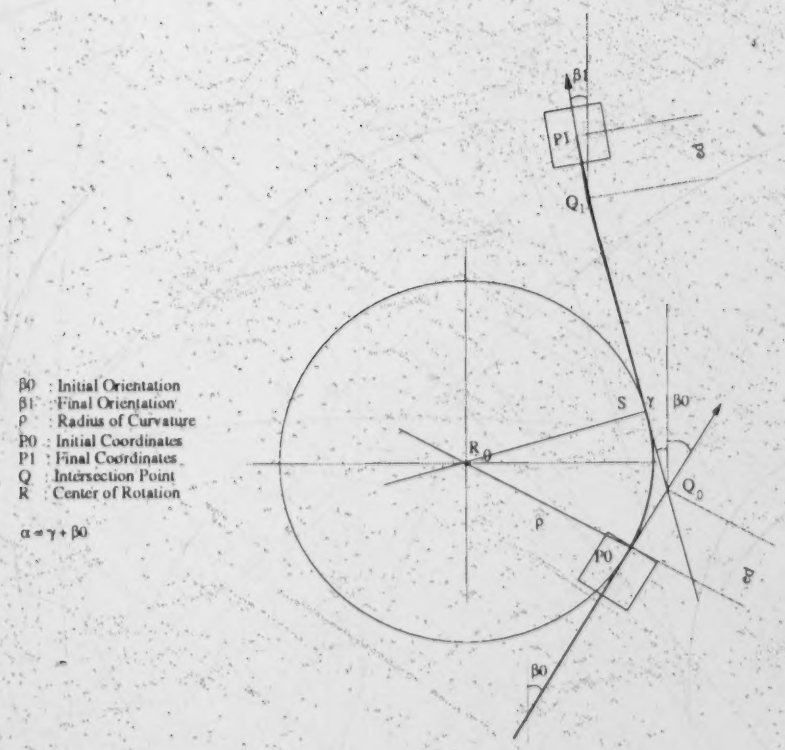

Figure 4.10: Details of a curvature

and the radius of curvature will be:

$$\rho = \left|\overline{P_0 R_0}\right|$$

The two-tuple (α, ρ) will be calculated for all segments with non-zero curvatures.

4.5 Motion Control

Having solved the kinematic problem, in this section we shall analyze the motion control of the robot.

4.5.1 Setting Wheel Velocities

The robot achieves a desired motion by setting the speed of the two independent wheels.

The types of commands that the robot can execute are

1. Straight line motion: the robot moves along a straight line between two points, the speeds of both wheels are $v_r = v_l \neq 0$.(see Figure 4.11a)

2. Turn motion: This type of motion causes the robot to move along an arc of a circle of radius ρ as shown in Figure 4.11b. The target speeds of the right and left wheels are $\frac{\rho+d}{\rho}v$ and $\frac{\rho-d}{\rho}v$ respectively. The sign of ρ designates the direction of the turn (clockwise or counter-clockwise).

By a sequence of straight line motion and turn motion commands a trajectory defined in the previous sections can be achieved.

An example of a motion sequence is given in Table 4.1, the corresponding trajectory in Figure 4.12a, and the velocity profiles of the two wheels in Figure 4.12b. The acceleration of the wheels is handled by the robot.

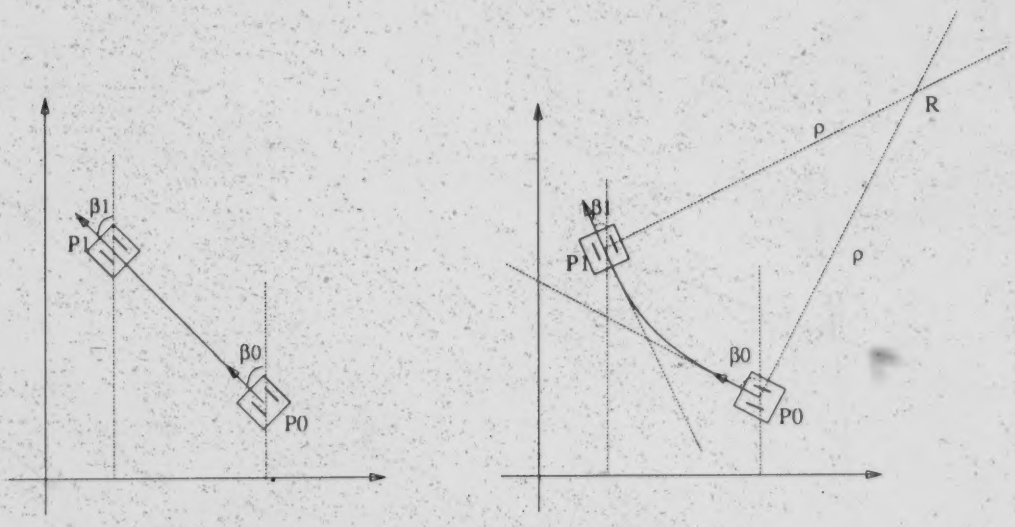

Figure 4.11: Possible types of motion

Motion Type	Time	Path
Straight line with v_1 for S_0	0 - t_0	$S(t_0, 0) = S_0$
Turn with ρ_0, v_1 for S_1	$t_0 - t_3$	$S(t_3, t_0) = S_1$
Straight line with v_1 for S_2	$t_3 - t_4$	$S(t_4, t_3) = S_2$
Turn with $-\rho_1, v_1$ for S_3	$t_4 - t_7$	$S(t_7, t_4) = S_3$
Straight line with v_1 for S_4	$t_7 - t_8$	$S(t_8, t_7) = S_4$
Turn with ρ_2, v_1 for S_5	$t_8 - t_{11}$	$S(t_{11}, t_8) = S_5$

Table 4.1: An Example Motion Sequence

4.5.2 Execution of the Trajectory

Over a short run, the encoders generally give an accurate information of the distance traveled. However, the orientation may be off by a few degrees. This error in

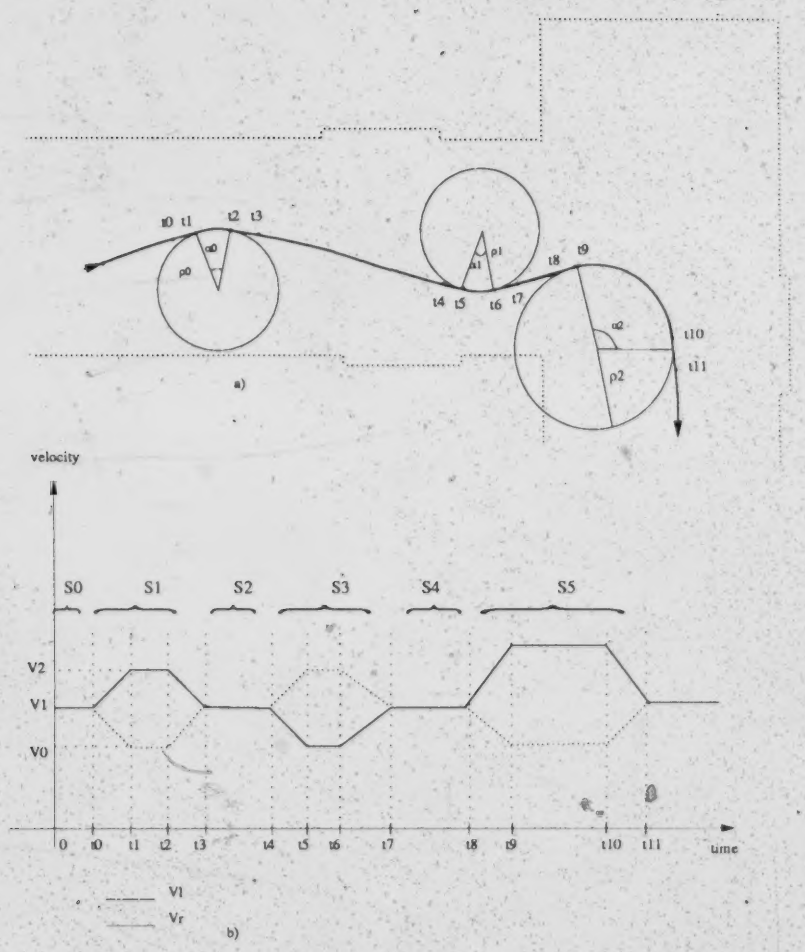

Figure 4.12: A sample motion and its corresponding velocity profile

orientation will lead to large Cartesian position errors in the direction orthogonal to the path. The analogy that is made by [Kriegman *et al.*, 1987] explains the concept briefly: "when a flagpole moves in the wind, its height will not change drastically, but the top of the pole may move parallel to the ground by many feet".

To determine the necessary modifications, changes in the robot's location are monitored. The corresponding path errors are computed by processing sensor data, as explained in Chapter 3 . Correction is done by computing a new path if the estimated error is larger than a threshold value. As new trajectories are computed, the previous ones are eliminated for the current sub-goal. Updating the trajectory at each correction time will give a control scheme similar to a *bang-bang control*[3]. Figure 4.13 illustrates a sample trajectory update.

Our trajectory generation function does not thoroughly address the unexpected obstacle avoidance problem in complex environments. Necessary action, when an obstacle is detected, is taken by the trajectory execution module. This action may either be regenerating the sub-goals or waiting until the obstacle disappears, depending on the geometry of the detected obstacle and the free space constraints.

4.6 Summary

In this chapter we have devised a trajectory generation method which has been developed on a basis of the previously introduced methods in the literature. The control issues in the chapter are system specific for the TRC Labmate. However, the functions are flexible to adopt another system. In addition, the path generation function is not bound to certain type of mechanics, so the method can be used with other types of mobile robot machinery as well.

[3]Bang-bang control is the control scheme which uses maximum deceleration for the motor in order to stop at a point.

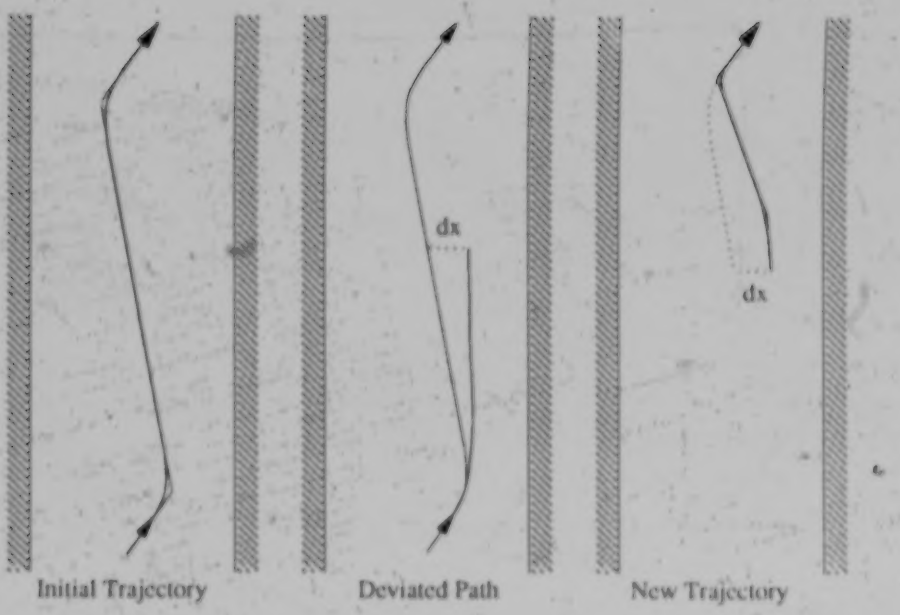

Figure 4.13: Updating a trajectory during navigation

The two basic aspects of the method are: its continuity and simplicity in computation. A real-time system can easily update the trajectory if a deviation from the expected path is detected with the help of a position estimation algorithm. The linear and curvatured segments used in the method are practically continuous. This provides a smooth trajectory execution and reduces possible errors that might have been caused by discontinuity.

Chapter 5

Implementation

In this chapter we will describe the application of the localization and trajectory generation methods described for the problem of autonomous mobile robot navigation. The methods are implemented on the robot system which has been described in Chapter 2.

The goals of the system are to provide a robust autonomous robot system in which sensors are used to achieve autonomy. The autonomy in our case is described as the ability to navigate in a known environment. The environment of concern is the corridors of one floor of an office building. Most of the implementation is devoted to manipulation, coordination and integration of sensory information. The domain of operation is restricted to a simple polygonal world. The trajectory generation module generates simple trajectories on a feedback mechanism between the sensory systems and the control variables of the robot. The system is programmed in high level language C.

In the following sections we will first describe the implementation of the communication within the system. Then we will explain the basic robot motion and sensor control routines. In Section 5.3 the implementation of the localization processes is

explained. The localization process involves observing the position information from multiple sensors and fusing this information into a final position estimate using the estimator described in Chapter 3. In section 5.4 we will discuss the trajectory generation and control methods and the feedback they impose on the system along with the perception method. Finally in section 5.5, we will discuss the experimental results and the problems encountered.

5.1 Communication

The communication of the system is done physically through two RS-232 lines with a baud rate of 9600 bps as shown in Figure 2.3. The Sun workstation is the main computational power that coordinates the whole system. The implementation of the communication is designed to allow modularity and flexibility for future development. The structure of the communication is similar to read-write operations in UNIX. A common library of functions to transfer particular data structures between named processes has been developed. This library consists of the basic communication routines between the host (Sun workstation) and the TRC mobile robot base and the proximity system. The library includes general-purpose routines like `sendbyte()`, `receivebyte()`, `handshake()` etc., and the built-in functions such as `enablesensor(sensorno)`, `point_to_point_go(dist)` etc.

The messages between the coordinator and the controllers (motion and sensory controllers) follow a fixed format, with a header comprising the destination tag and the message length, message identifier and the message body. To provide flexibility, the length of a message is not fixed. A variety of commands have been implemented including motion control, sensory control and information handling commands, using this communication scheme.

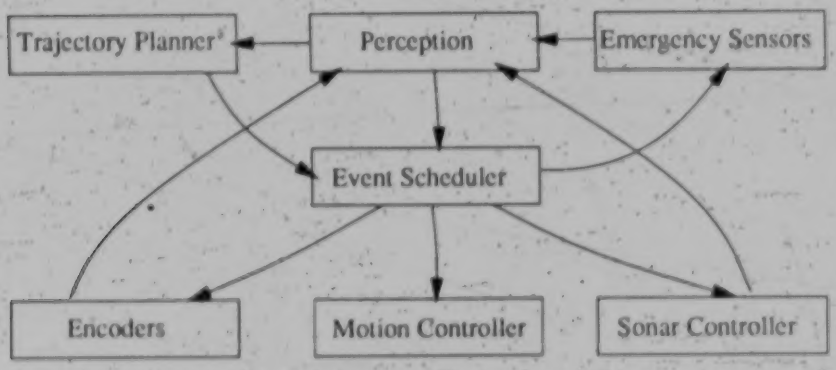

Figure 5.1: The communication between processes

5.1.1 Scheduler

In order to provide quick reactions to external events, the system is interrupt driven. The Unix library function `setitimer()` [Sun, 1989] initiates the timer to generate a periodic `SIGALRM` signal each time the period expires. Decrements in the timer are made in real time without considering if the system (CPU) may be running on behalf of other processes. The interrupt vector is set to the routine `AlarmHandler()` which temporarily disables the timer and activates `event_scheduler()` to execute the scheduled commands and the communication with the servo controller. The communication with the servo control system involves reading the encoder information and issuing scheduled commands.

The scheduler is responsible for scheduling the events and their execution. It is the main coordinator of the system which receives scheduling requests from the perception module and the trajectory planner. The perception module sends requests which include the scheduling of emergency actions. The trajectory planner sends requests for scheduling motion commands of the trajectory, and cancelling some previously scheduled events. The interprocess communication of the program is shown in Figure 5.1.

The structure of an event is as follows:

```
struct t_event {
                void (*callto) (); /* the function to be activated */
                int parcount; /* no of parameters */
                long **parameters; /* parameters to the function */
                double s_time; /* starting time */
                struct t_event *next; /*next event in the list */
                } t_event;
```

Each entry in the list contains the details of an event, i.e. the parameters and the time that the action is going to be executed.The `s_time` field is also used for priority assignment, as the scheduler scans the list and activates the event with the earliest starting time, if its starting time is smaller than the system time.

The alarm time set for each interrupt is $100msec$. The main program is interrupted after it is executed for $100msec$ and the `AlarmHandler()` takes over. `AlarmHandler()` activates `EmergencyCheck()` function at the first place, which controls exceptions like bumper contacts, power failures etc. After the emergency checks, `UpdateReckon()` is run to update the dead reckoning information. Then `EventScheduler()` is called which activates the first event in the event queue. After the event's execution, if the time elapsed since the beginning of the interrupt is less than $70msec$, another event is scheduled. At the end of the interrupt the alarm time is set to $100msec$ again and the main program resumes execution. The above scheme is illustrated in Figure 5.2. Using this scheme the average execution time for the routine `AlarmHandler()` is about $80msec$ on a SPARCstation SLC.

The communication with the servo controller and the proximity system is also activated by the scheduler. This avoids the blockage of any communication (especially emergency commands). A deficiency of this method is that an emergency case cannot be realized as soon as it occurs. The maximum delay time is about $170msec$. However,

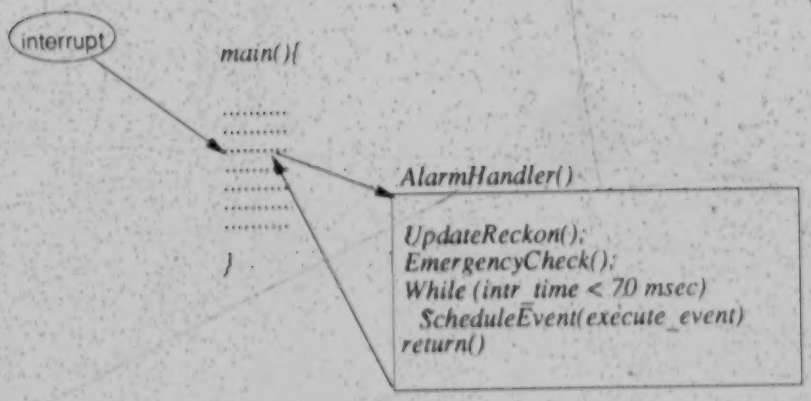

Figure 5.2: The interrupt scheme

the servo control system is able to perform an emergency stop by itself as soon as an emergency occurs, in order to ensure that catastrophic failures are properly prevented.

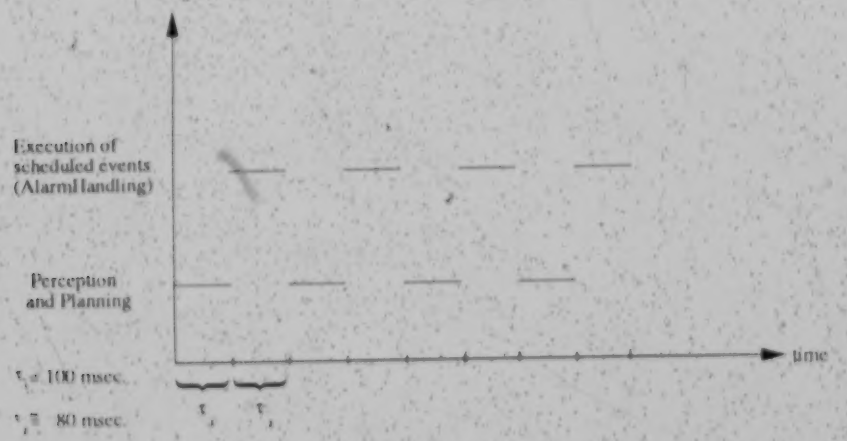

Figure 5.3: The scheduling mechanism

5.2 Motion and Sensory Control

5.2.1 Motion

The trajectories generated by the trajectory planner require two types of motion,

straight traveling and turn motions. The availability of a `jog()` command in the TRC mobile robot software [Tra, 1989b] allows us to combine a certain turn velocity with the current straight line velocity to perform a turn. The servo control system of TRC implements three modes of operation. Specifically,

Go mode moves the robot in a straight line at the current velocity setting. This motion continues until another mode command is issued.

- `point_to_point_go(dist)` performs a point to point travel for a given distance. The parameter `dist` is the amount of distance to be traveled in millimeters.
- `go(velocity)` The robot starts moving in a straight line motion with a velocity equal to `velocity`. The movement continues unless another command is received.

Jog mode Superimposes a continuous turn rate atop an existing forward velocity. Turning continues until a *go mode* or another *jog mode* command is issued.

- `jog(rate)` This command puts the robot in `jog` mode. The single parameter `rate` is the turn rate in degrees/second.

Turn mode The robot performs a discrete turn in this mode with the desired amount of turn.

- `ptp_rel_turn(degrees,radius)` This command is used to perform a turn of `degrees` with a radius of curvature equal to `radius`. The amount of turn is relative to the initial orientation. The robot stops after it completes the turn.

The velocity of the robot is set to $300mm/sec$ in our experiments. This is a pragmatic value which tolerates the response time of the perception system to possible errors within the corridors.

To illustrate how a motion trajectory can be accomplished using the above commands, refer to the example in Table 4.1. The necessary commands for this motion will be:

```
point_to_point_go(length(S0));
jog (a0/(t9-t0));
point_to_point_go(length(S2));
jog (a1/(t7-t4));
point_to_point_go(length(S4));
jog (a2/(t11-t8));
```

5.2.2 Exception handling

There are a number of failures that the TRC mobile robot base can detect. These may be listed as

- A bumper contact
- A current overload
- An encoder failure
- A blown fuse

When any of these failures is detected, a certain condition bit of the robot controller is set. The command `readstatus(statbyte)` reads this register and if there exists an emergency condition, activates the routine `EmergencyHandler()` immediately. This command is issued at the beginning of each interruption of the communication module, thus the maximum time between the detection of and reaction to an exceptional condition is limited by approximately 180ms (See Figure 5.2).

5.2.3 Sensor Control

The system employs two types of sensors, the wheel encoders and the sonar proximity system. Encoders are simply counters which monitor the angular displacements of the wheels. In order to find the position change of the base coordinate of the robot, the encoders are read through the serial line by the command `ReadEncoders(rightreg,leftreg)`. The encoder registers are 16 bits and contain the amount of distance traveled by each wheel in millimeters. These values are mapped to the Cartesian coordinate frame through the kinematics of the robot, using equation 3.12.

The 16 ultrasonic transducers of the proximity system are controlled by two sonar ranging modules [Tra, 1989a]. The sensors are numbered 0 through 15. The first module controls transducers 0 through 7 (first set), and the second module controls transducers 8 through 15 (second set). The sonar sensors are scanned individually in a round robin manner. Since the time required for a sonar scan is limited by the speed of the sound and by the triggering delays of the relays controlling the sensors, each scan may take tens of milliseconds. This constitutes a bottleneck for the system which limits the period in which a localization is made. An update flag is set for each sensor whenever a new reading is made. The flag is reset when the information is retrieved. This provides continuous scanning of the sensors, while the host is busy with some other process.

The exact location of the robot when an echo is received has significant importance as the reflection point will be computed relative to this location. However, the proximity system does not provide a parameter (time or position) associated with each echo reception. It is not possible to figure out the exact value of this parameter unless the robot is stationary during the sonar reading. We assume the location of concern to be the estimated position of the robot at the instant when the sonar register is read. Each reading is assigned the position kept by the encoders. The

encoder positioning is used for this purpose which is updated every 180 msec. This is the best estimate we have at the time of the reading. The encoder positions are kept in global variables, `ENCX`, `ENCY` and `ENCTHETA`. With the current system parameters[1] this assumption may cause an error less than 30 mm in each reading.

There are many ways to control the sensors. One way is enabling one sensor at a time and then waiting for a response. In such a case the error in each reading decreases to less than 15 mm but the total time for the complete scan of the 16 sensors goes up to approximately 3.2 seconds from 1.3 seconds because of the delays in the relays. This is a more critical factor than the error as the robot moves about 1 meter in 3 seconds, so the above method is not used. Another way to control the sonar sensors, which we actually use for our system, is to enable all sensors and poll for echoes in a round robin fashion. As a result the position uncertainty associated with an echo is increased but the total delay is minimized.

5.3 Localization

The perception module is composed of three main components (see Figure 2.7), the sonar sensor localization module, the encoder localization module and the data integration module which fuses the outputs of the two.

5.3.1 Sonar Sensor Localization

The sonar transducers mounted on the robot base are represented by an array which consists of their positions with respect to the robot's coordinate frame and their calibration factor. Sonar readings are represented by a structure as follows

[1] Interrupt time = 100msec, Velocity = 300 mm/sec

```
struct sonarpoints {
        double              robotx, roboty, dist, robottheta;
        int                 sensorno, update; /* sensor no and flag */
        int                 comment; /* to be used later*/
        struct mapnode      *mnode; /* associated mapnode */
        struct sonarpoints *next; /*next reading */
        }                   sonarpoints;
```

where `robotx, roboty, robottheta` are the approximate location identifiers of the robot when the sonar echo is received. `dist` is the reading of the sensor in mm, and updates the flag indicating whether the sensor has been updated since the last reading or not. `mapnode` is a segment of the a priori world map from which the reading is found to be reflected.

Sensor localization is done after each scan of the 16 sensors and it will be referred to as a *complete scan*. The equations 3.13 and 3.14 are solved for each consistent data point. Each point is searched through the world map [2] for a candidate reflection point. Each node of the world map has the following structure

```
struct mapnode          {
        int             nodeid; /* encoded id */
        struct line     nodeinfo; /* physical information*/
        struct mapnode  *brother, *sibling, *parent; /*pointers*/
        }               mapnode;
```

The `nodeid` field is used to group the nodes belonging to a certain segment. The info field has the following structure

[2] The world modeling was explained in section 2.2.1.

```
struct line             {
        double          a, b, c; /* coefficients*/
        long            length, midx, midy; /* location */
        double          adjacency; /* thickness */
        /* in the form y/a + x/b = c */
        }               line;
```

which consists of the information of a line segment plus the adjacency field indicating the thickness of this line in order to span the physical locations of its siblings, if any (See Figures 2.9, 5.4).

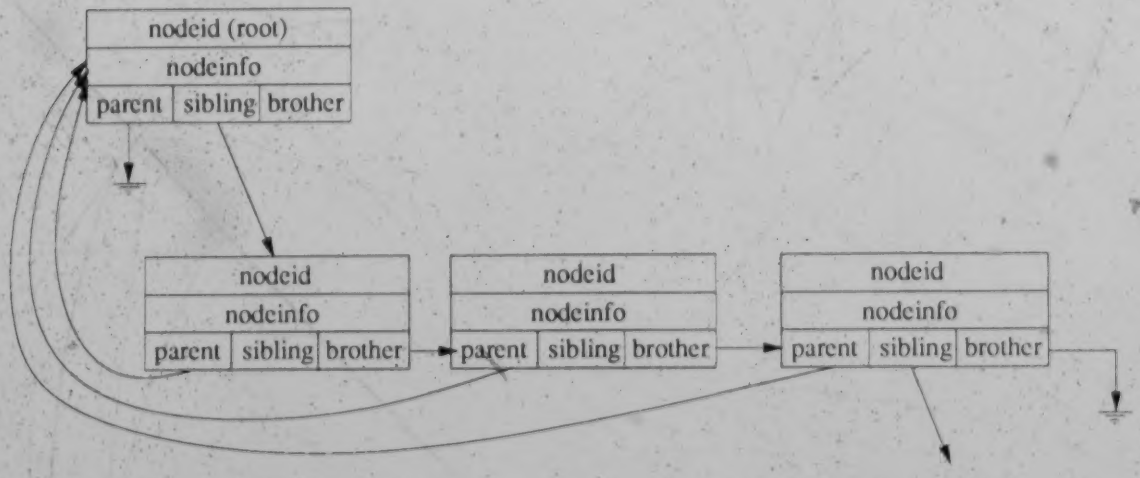

Figure 5.4: The world map

The search through this map is done with a recursive function `search(rootnode,x,y)`. The search algorithm is shown at figure 5.6. The algorithm searches for a candidate segment of reflection by traversing the world model in a depth-first manner. The consistency tests described in section 3.3 (visibility, proximity and cone of reflection tests) are applied by the function `consistent(mapnode,x,y)`.

To show the search program in realistic terms, consider a sample scan in the corridor which is shown in Figure 5.5. The point P from sensor 4 satisfies the con-

sistency tests for both of the lines l_1 and l_2. It is associated with l_1 as the distance between them is smaller.

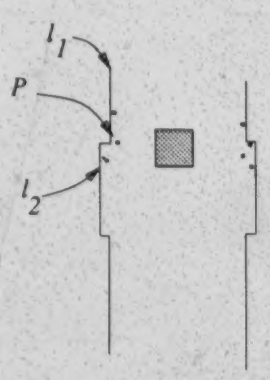

Figure 5.5: Associating a point with a line

```
mapnode
* search(rootnode, x, y, slist)
      mapnode      *rootnode;
      double       x, y;
      mapnode      *slist;
{
      mapnode      *node;
      node = rootnode;
      while (node)
         if (consistent(node,x,y))
            if (node->sibling)
              addlist(slist,search(node->sibling,x,y,slist);
            else addlist(slist,node);
            node = node->brother;
      }
            return (slist);
}
```

Figure 5.6: The search algorithm

After the inconsistent points are eliminated, the remaining consistent readings are used to solve equations 3.13 and 3.14 to get a position estimate (x, y) using the weighted average of the resulting solution from each sonar sensor. The orientation is computed by the routine `WeightedLeastSquare(sonarlist,sonarweights)` which uses the weighted least square fitting method described in section 3.5 The weights are computed using the methods described in section 3.2.

In order to use as much data as possible for a single segment, readings from the doors, which are a few centimeters from the wall, are transformed to the wall's

position. This is illustrated in Figure 5.7a. The readings marked with d in the figure are transformed to the left by an offset equal to the depth of the door in the wall. Figure 5.7b shows the line segments formed when this technique is not used. The segments to be combined together to form a line are identified on the a priori world map by encoding their `nodeid`'s.

Figure 5.7: Forming lines by point transformation

5.3.2 Encoder Localization

The function `UpdateReckon()` reads the encoder values of the wheels and updates the encoder positioning using equation 3.12. The routine is activated at the beginning of each interrupt after emergency checks.

The following example illustrates the encoder localization:
the readings from the left and right encoders have the values $E_{l_t} = 154mm$ *and* $E_{r_t} = 162mm$ where the previous measurements were $E_{l_{t-1}} = 141mm$, $E_{r_{t-1}} = 140mm$ and $P_{t-1} = (x, y, \theta)^T = (105mm, 4562mm, 1.40°)^T$. The distance between the wheels is $300mm$.

We calculate $\Delta E, \Delta P$ and P_i as follows

$$\Delta E_r = 22, \Delta E_l = 13$$

$$\Delta P = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix}$$

$$\Delta P = \begin{bmatrix} Sin(\theta_{i-1})\frac{\Delta E_r + \Delta E_l}{2} \\ Cos(\theta_{i-1})\frac{\Delta E_r + \Delta E_l}{2} \\ atan(\frac{\Delta E_r - \Delta E_l}{wheeldist}) \end{bmatrix}$$

$$\Delta P = \begin{bmatrix} 0.8551 \\ 34.9595 \\ 0.0300 \end{bmatrix}$$

$$P_i = P_{i-1} + \Delta P = (155mm, 197mm, 1.43°)^T$$

Figure 5.8 shows a sample transformation of the robot from P_{i-1} to P_i.

Figure 5.8: Encoder geometry.

The weights assigned to each sensor are computed after a complete scan. The method uses the last k scans in order to account for the recent constraints of the environment. The value of k used in our implementation is 10.

Two arrays, `samplemean[16]` and `samplecovariance[16]`, for the sonar transducers are updated after each complete scan. They are computed using the equations 3.4 and 3.6. The error term of the last k readings is found using the position estimates found at their localization steps by subtracting the reading from the estimated rellection point. As mentioned before the `robotx`, `roboty` and `robottheta` fields of

each sensor data of the last scan are updated using linear approximation after each localization. These will be used in the next weight computation.

The encoder weights are also computed from the last k location differences at each localization step where the encoder position and the estimated position are kept in an array which is updated after each localization.

5.3.3 Data Fusion

The maximum likelihood estimator which was explained in section 3.6 fuses the data using the covariance matrices of the errors of the two sensory systems. Sample covariances which was explained in section 3.2 are used to compute the sample covariance matrices of the sensors. The three arrays, `SonarLocation[k]`, `EncoderLocation[k]` and `Estimate[k]` are used to compute the sample mean and covariances using equations 3.4 and 3.6.The final position estimate is computed using equation 3.25. The library utility `InverseMatrix(matrix,size)` is used to evaluate this equation. After the fusion process, the three arrays, `SonarLocation[k]`, `EncoderLocation[k]` and `Estimate[k]`, are updated so that they will contain the last k localization results.

The critical resource for final localization is the input from sonar proximity system. A complete scan of the sensors takes an average of 1.3 seconds which constitutes the bottleneck for sonar localization, whereas an encoder localization is done in every 180 msec. Thus data fusion is made when the position information from the sonar perception system arrives.

This fusion architecture can be extended in order to include more sensory systems. To introduce a new sensory system into the program, it is necessary to specify the characteristics of the sensor, declare its structure and implement its localization method. An example of final sensory fusion will be given in section 5.5.

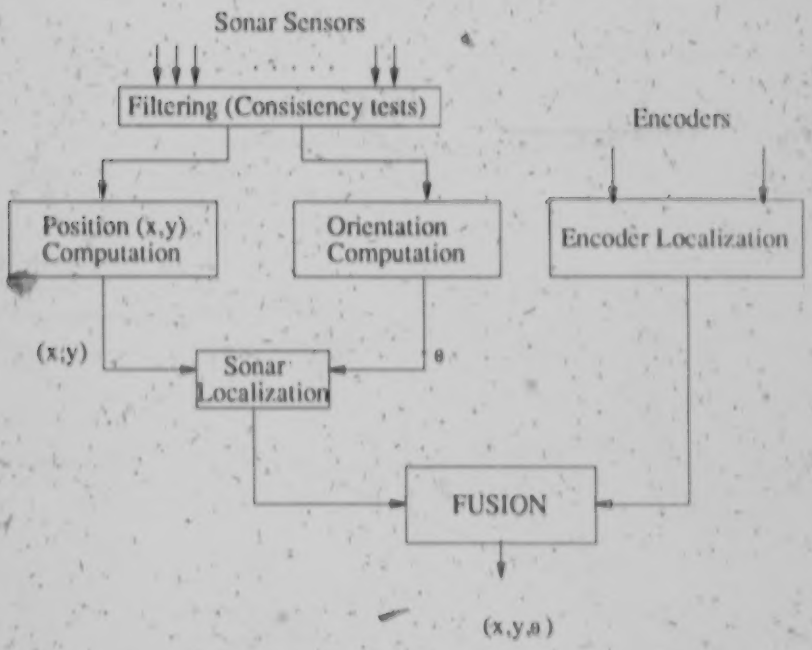

Figure 5.9: Data flow of localization

5.4 Trajectory Generation and Control

At present, the only task that our robot can perform is to travel between locations. Thus, a goal is represented as one or more positions in the free space. The trajectory generated for this goal is represented as a list of nodes where each node describes a motion segment. The structure of a node is as follows.

```
struct traj_node    {
       double       velocity; /* the velocity during this motion */
       double       startx, starty, starttheta; /*start point*/
       double       destx,desty,desttheta; /*end point*/
       byte         straight_motion; /* flag for no turn */
       double       turn_radius; /* the radius of curvature (if any) */
} traj_node;
```

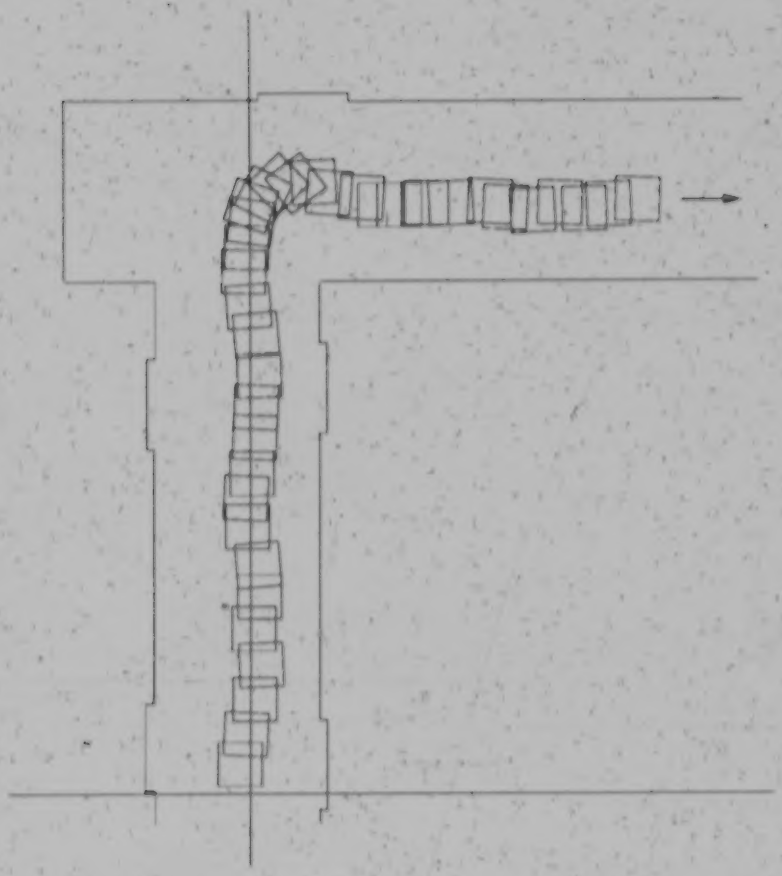

Figure 5.10: A sample run within the corridors

The motion described by the above structure can be defined as "traveling at a given speed between the start and end points with a given radius of curvature". A straight line motion is identified by the variable `straight_motion`. If `straight_motion` is true then the `turn_radius` is not considered.

The trajectory list is generated using the methods described in chapter 4. The trajectory generation is done at the beginning of navigation, and the successful execution of this trajectory is controlled throughout the program by `TrajCheck()`. The scheduler activates `TrajCheck()` which scans the trajectory list for the next motion to be executed once the current one is completed. Completion is detected by checking the current position with the destination points. In that case the completed node is

disposed and the motion control parameters of the robot are set to the next node's specifications. The trajectory controller also checks the deviations from the planned trajectory. This is done only during straight traveling motions or turns with a small curvature (less than $5°/sec$), because the sensor data is unreliable during turn motions with considerable curvatures. When a deviation over a certain threshold is observed, corrections on the trajectory list is done by modifying the necessary trajectory nodes and introducing additional ones. Figure 5.11a shows a deviation from the path. Figure 5.11b shows the correction of `TrajCheck()`, the two nodes **I** and **II** are replaced by new nodes **I, II** and **III**. The threshold values are set accordingly with the response level of the servo system, for example the smallest turn that can be realized is $2°/sec$.

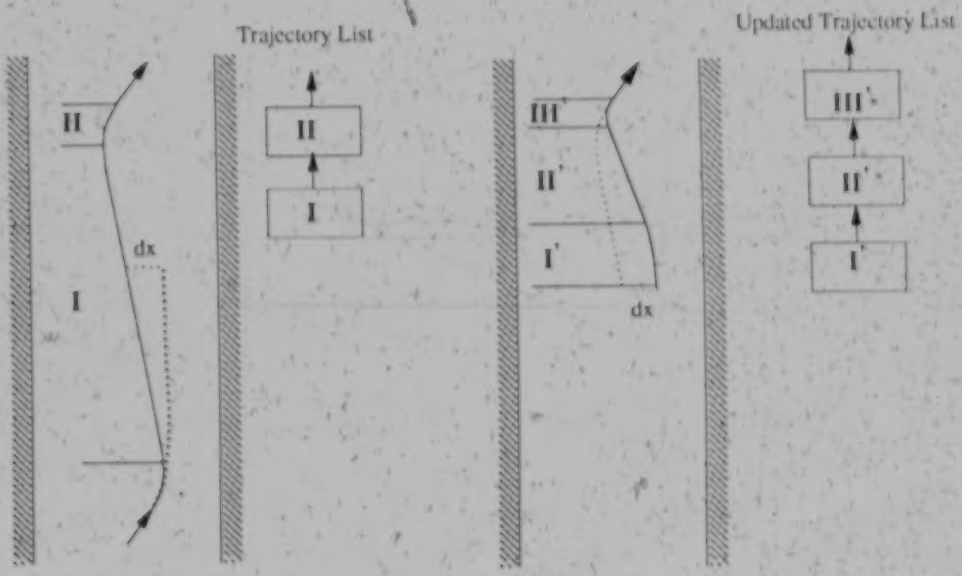

Figure 5.11: Correction of a trajectory by overriding the last two nodes

The constraints of the environment we realized in our implementation do not require complex or precise maneuvers to reach a destination point. Because of this, the velocity can be set high during navigation as much as the sensitivity of the error in positioning allows it to be. The control parameters, like velocity and turn rate, are

set accordingly and have maximum values of $300mm/sec$ and $30°/sec$ respectively.

There are two cases in which the robot faces an unexpected situation, and the worst outcome may result in a collision. These two cases and their remedies are:

1. The robot recognizes an unexpected obstacle in front. The action taken is to stop and wait for the obstacle to move out of its way. If it does not move within a certain time, a new trajectory, considering this obstacle, is generated (in our case in the opposite direction), and executed.

2. If one of the sides of the robot is very close to a wall, the robot changes its mode from *achieve-goal* to *relocate*, i.e. it tries to re-position itself by backward and forward motion, temporarily abandoning the goal.

Another function `TrajCheck()` provides is decelerating as the robot approaches turns with an angle greater than 30°. This is done in three steps by scheduling three `SetVelocity()` events with starting times computed using the approach velocity to the point.

5.5 Experimental Results

The performance of the robot depends on the performance of the sensors which depend on environmental constraints. The environment in our experiments was one floor of an office building. As indicated before, the two types of sensors employed in the implementation were optical wheel encoders and sonar proximity sensors. Optical encoders are sensitive to surface characteristics of the floor, and sonar sensors are sensitive to flat, vertical surfaces within the environment.

In the experimental environment, the translational accuracy[3] of the wheels'

[3]When the robot travels in a straight line motion.

encoders is about 95%. However, the rotational accuracy[4] is about 80%. The latter poses a serious problem as it leads to considerable disorientation of the robot, which in turn leads to large orientation errors. For example, with a velocity of 300mm/sec, a disorientation of 15 degrees may cause the robot to crash into a wall in 5 seconds.

The structure of the walls provides a convenient reflection source for the sonar sensors. Usually 9 of the 16 sensors receive consistent sonar echoes when the robot is inside the corridor. The accuracy of the sonar readings depends on the distance from the object and the reflection angle[5]. For example reflections with a zero degree of reflection angle gives an accuracy of 95% whereas this accuracy goes down to 75% for 15 degrees. Experiments showed that the maximum distance for a recognizable object inside the corridor is approximately 10 meters. For larger distances we encounter multiple reflections (See Figure 3.3).

Turning a corner so far has been the most difficult segment of a trajectory for the robot to execute. This can be attributed to three factors. First, the encoders are error prone during turn motions. Second, the number of consistent reflections from the sensors is quite low (2-5 reflections). Third, the robot position associated with the sonar readings when the echo is received is also inaccurate due to the inaccuracies of the encoders. The positioning errors caused by above factors may reach a maximum of 500mm and 15 degrees. The correction is done by decelerating the robot and executing relocalization procedures after the turns. This error may be avoided by introducing a sensor which does not fail at these locations (e.g. a CCD camera).

The overall performance of the system suffices well for our initial goal of continuous navigation through the corridors. Figure 5.12 shows a sample navigation. The real path is marked by a dotted line and the robot's position estimation is marked with a solid line. As seen in the figure, the robot recognizes the deviations from the path and corrects them accordingly. Figure 5.13 illustrates another path with

[4]When the robot performs a turn at standard speed (v=300mm/sec, w=20°/sec).
[5]The angle that the sonar beam makes with the normal to the reflection surface.

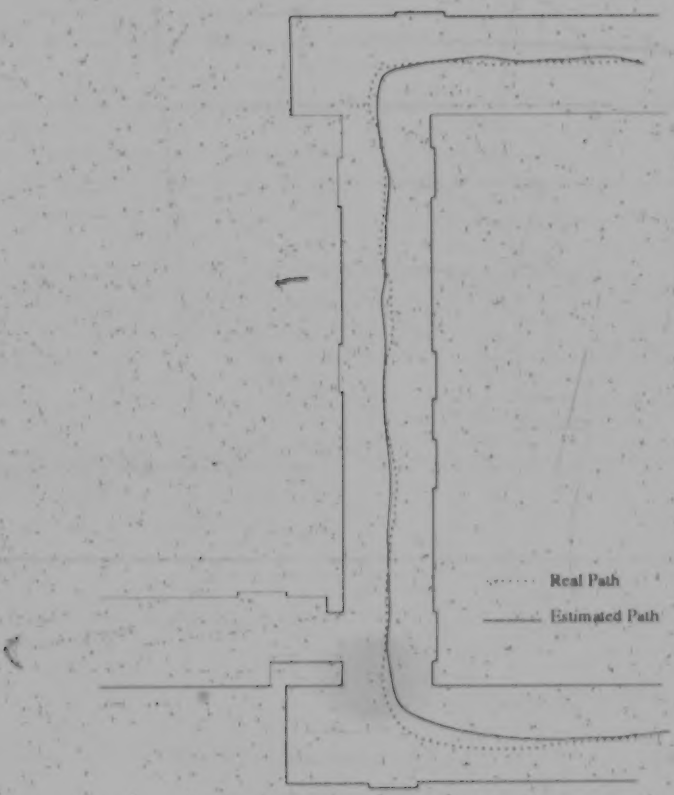

Figure 5.12: Real and estimated path

the error-time plots. The absolute error in the position parameters (x, y, θ) show a fluctuating pattern. However, they do not exceed a limit, unless a drastic failure occurs.

In Figure 5.13 we analyze localization errors along the three coordinates. In Figure 5.13b, the error increases between 20th and 60th second. This is because the robot is in corridor b where the walls are parallel to the x axis. Thus, there are not much reflection points to obtain the x position of the robot. The corrections seen in the graph are done by means of reflections from door gaps. The corners of the door gaps almost always give a reflection once they are inside the cone of reflection of a sensor. The error in x coordinate in this region rises up to 250 mm. However, this is not a critical error as the robot is traveling along the x axis. A similar situation can

be observed in Figure 5.13d for the x axis in corridor a. The turn points at 25th and 60th second result in high errors in all the three plots, because of the aforementioned reasons.

One critical error for the robot is the relative positioning error with respect to the walls. It is critical as it may cause the robot to collide with the wall. This error in our test trajectories has a maximum value of 60mm. Another critical error is the orientation inaccuracy which has a maximum value of 5 degrees within the corridors but may go up as high as 15 degrees when a turn from one hallway to another is involved.

The response time of the robot to a critical error is approximately 2 seconds. As the error increases, the response time approaches 1.3 seconds which is the time for a single localization process. This is sufficient considering the inaccuracies of the sensors used. With the current response time and accuracies, the robot is able to navigate at most 300 meters without colliding with an object.

Figures 5.14 to 5.19 show the x and y coordinates of the trajectory and the corresponding localizations of the sensors and the final estimate. Sampling is done in discrete intervals of 1 second. Figures 5.20 to 5.25 illustrate the errors in sensor localization in x and y directions, and the error in orientation. The sensors' errors are compared to the error in the final estimate. As seen in the figures, along the time axis, the error in the final estimation is usually biased to the sensor which has the smaller error. This result shows the validity of our uncertainty computation method and the fusion algorithm of sensor localization.

5.6 Summary

This chapter has discussed the implementation of the methods developed for

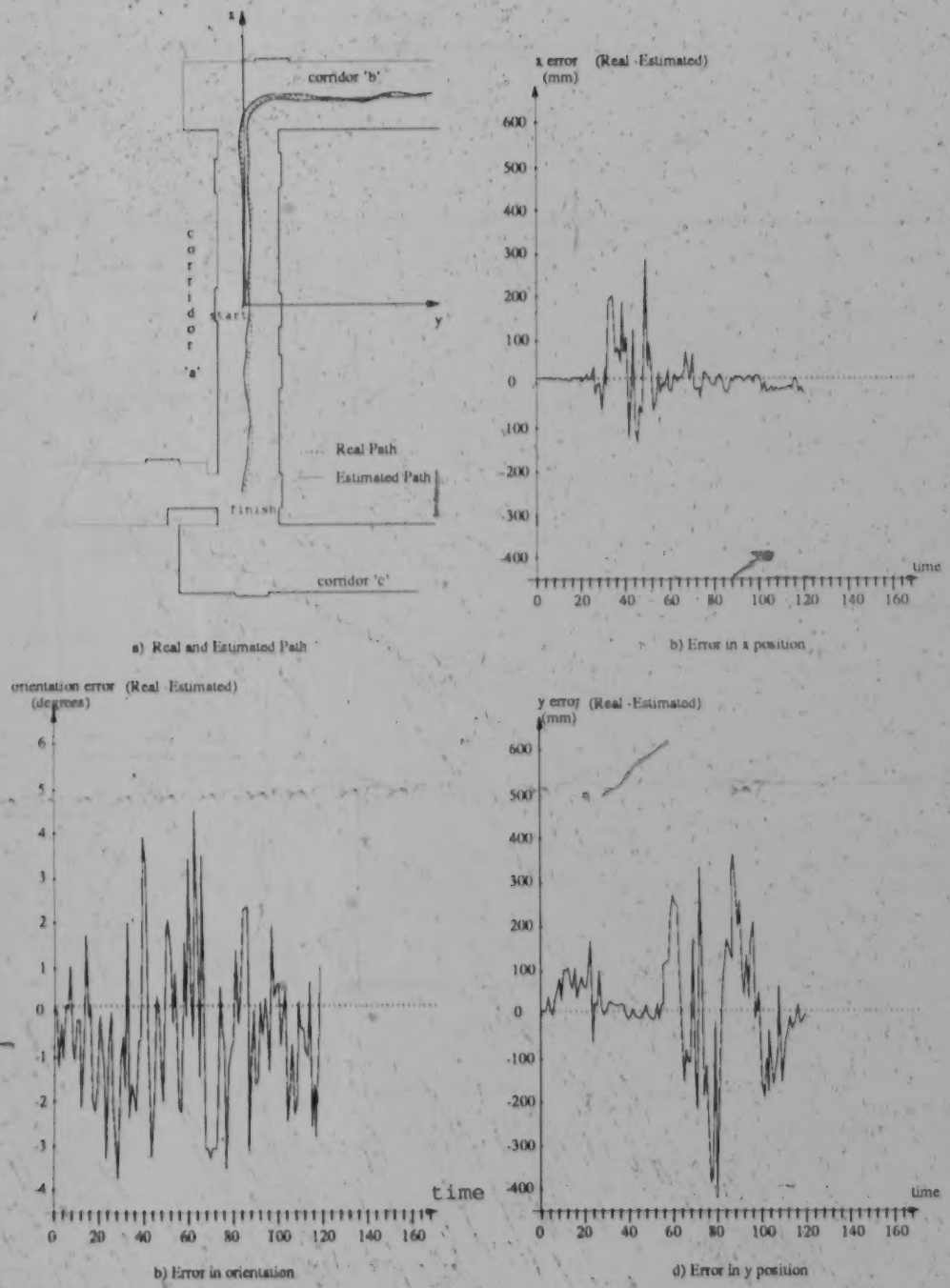

Figure 5.13: Localization errors along the path

localization and trajectory generation in the previous chapters. The implementation is done in C language to facilitate future modifications. Other software used in the implementation include the low level control routines for the TRC mobile robot and the proximity system which reside on eproms on their control boards. The resulting system is well functioning and it fulfills our initial requirements for the robot. The experimental results have shown the validity of our approach to robust robot localization using sensory fusion techniques. Without these techniques the robot performance will be considered seriously or completely unsatisfactory, if only one sensor system, either the sonar system or the wheel encoders, is used.

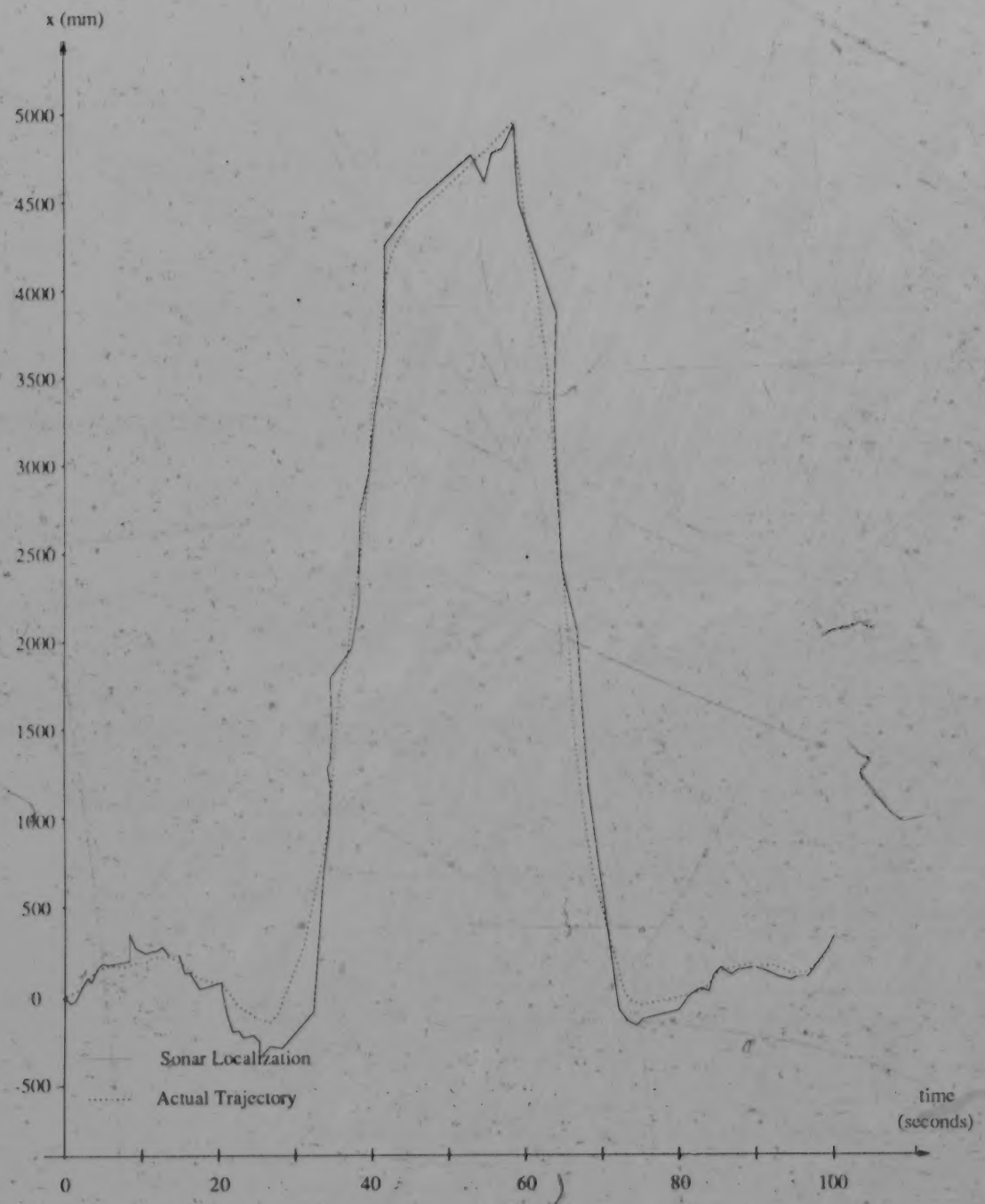

Figure 5.14: x coordinate of the actual location and the sonar localization

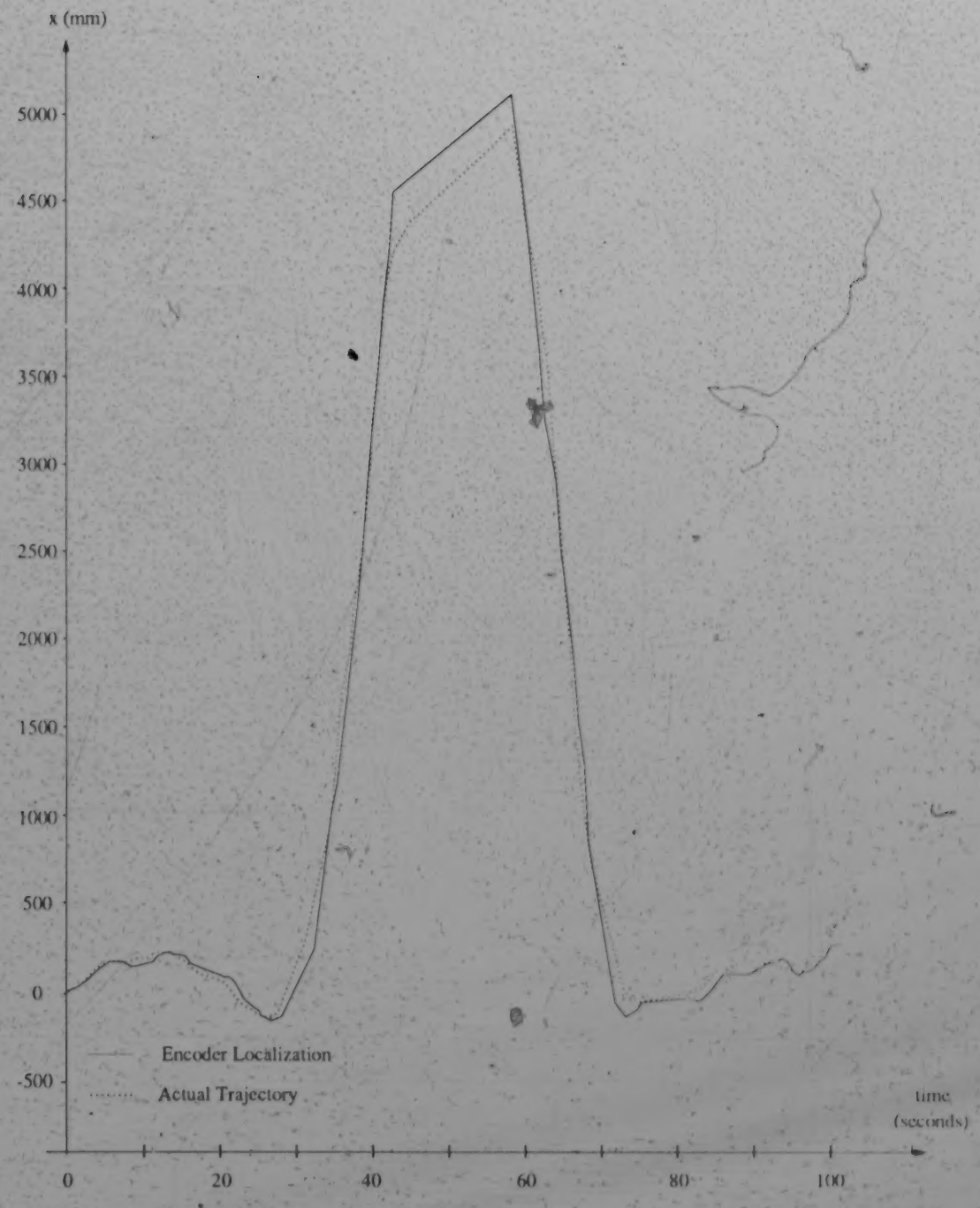

Figure 5.15: x coordinate of the actual location and the encoder localization

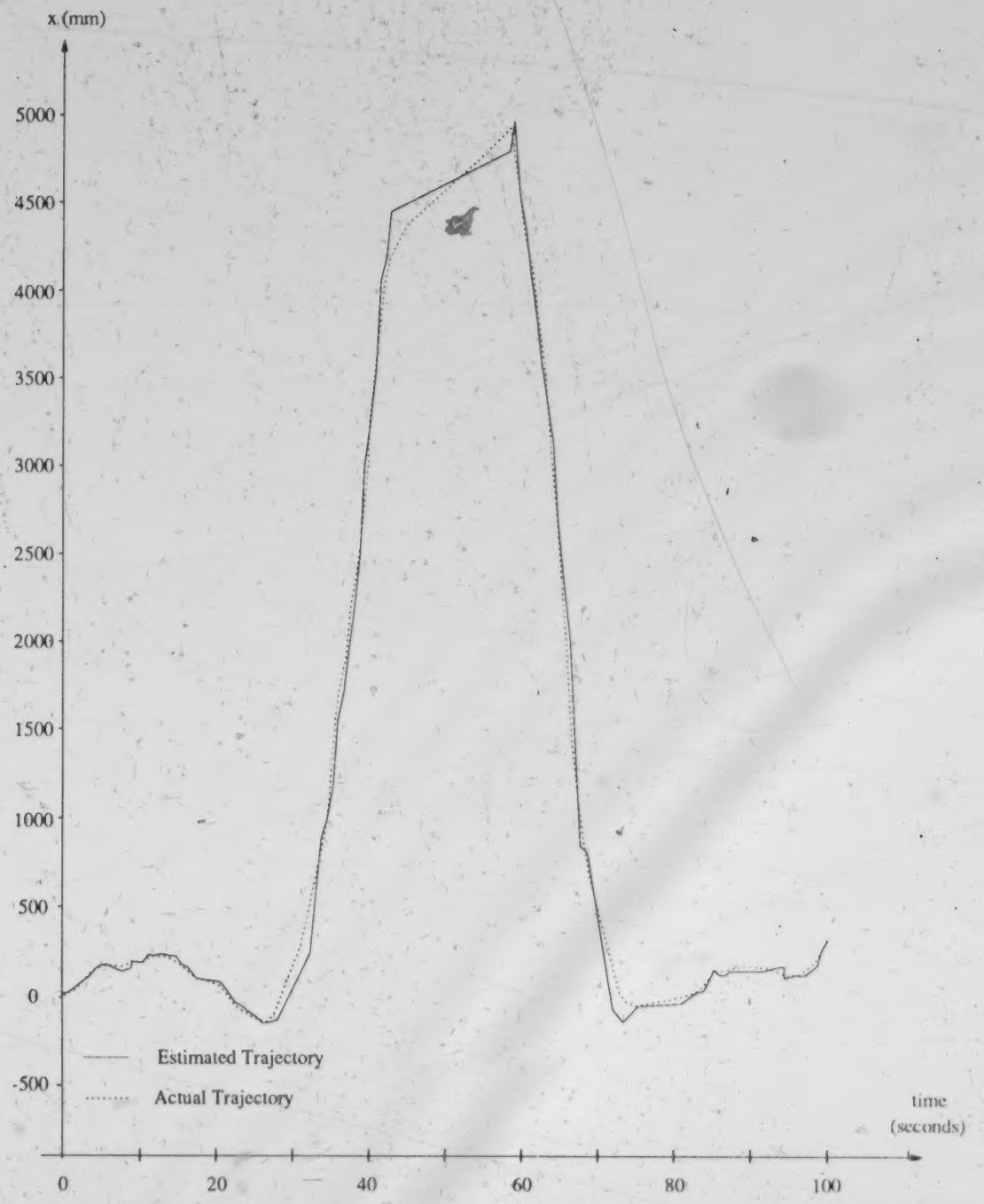

Figure 5.16: x coordinate of the actual and the estimated location

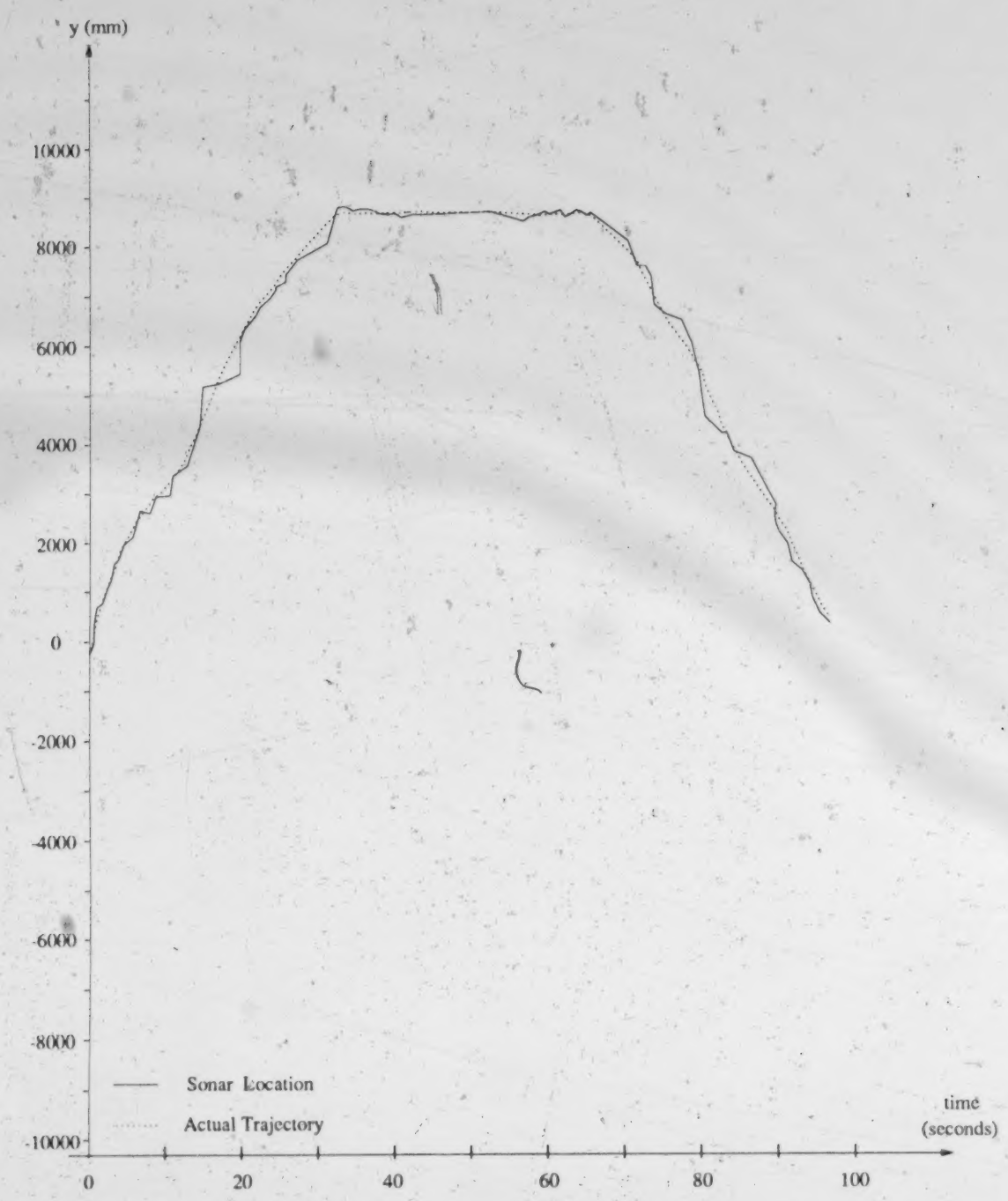

Figure 5.17: y coordinate of the actual location and the sonar localization

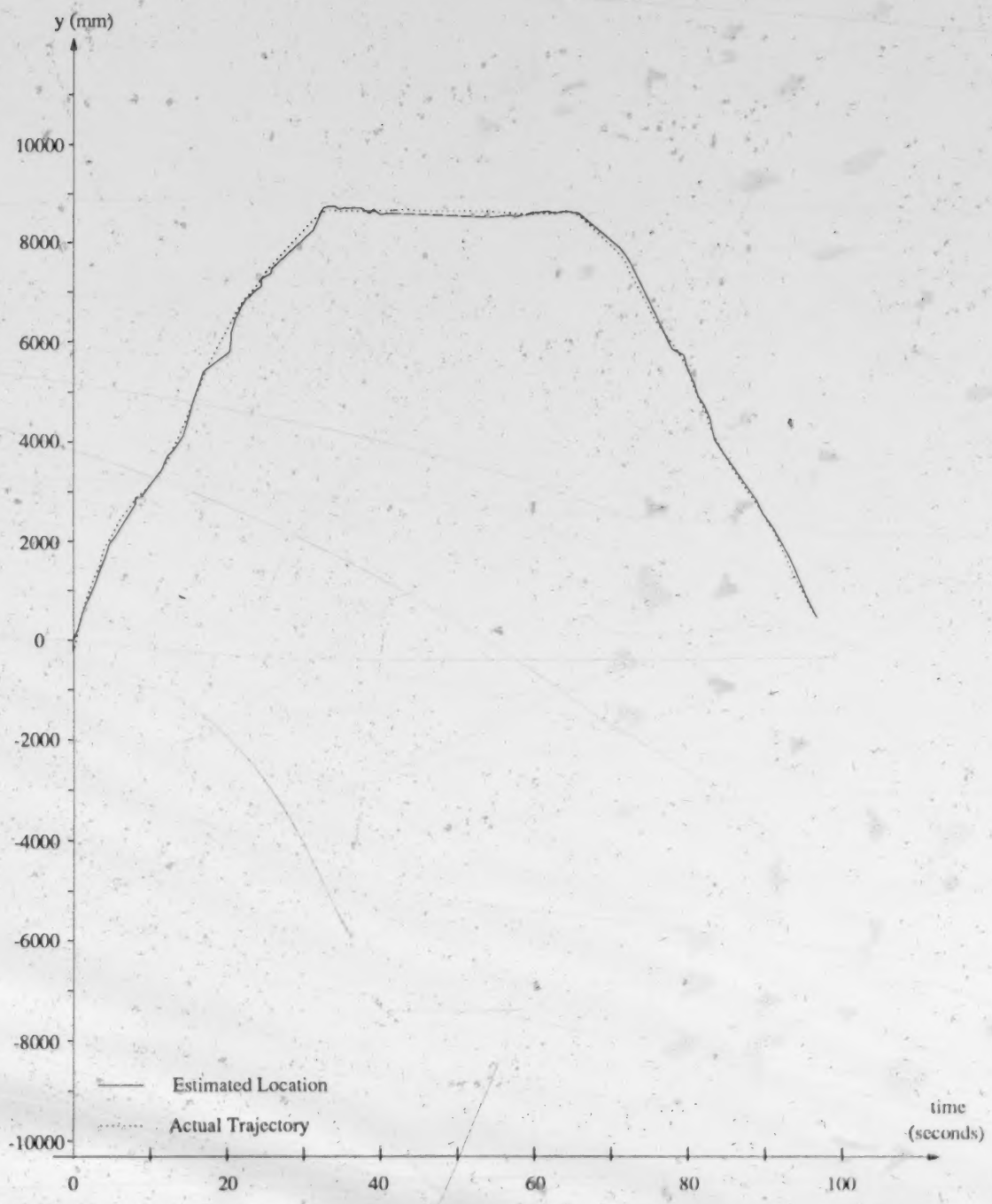

Figure 5.18: y coordinate of the actual location and the encoder localization

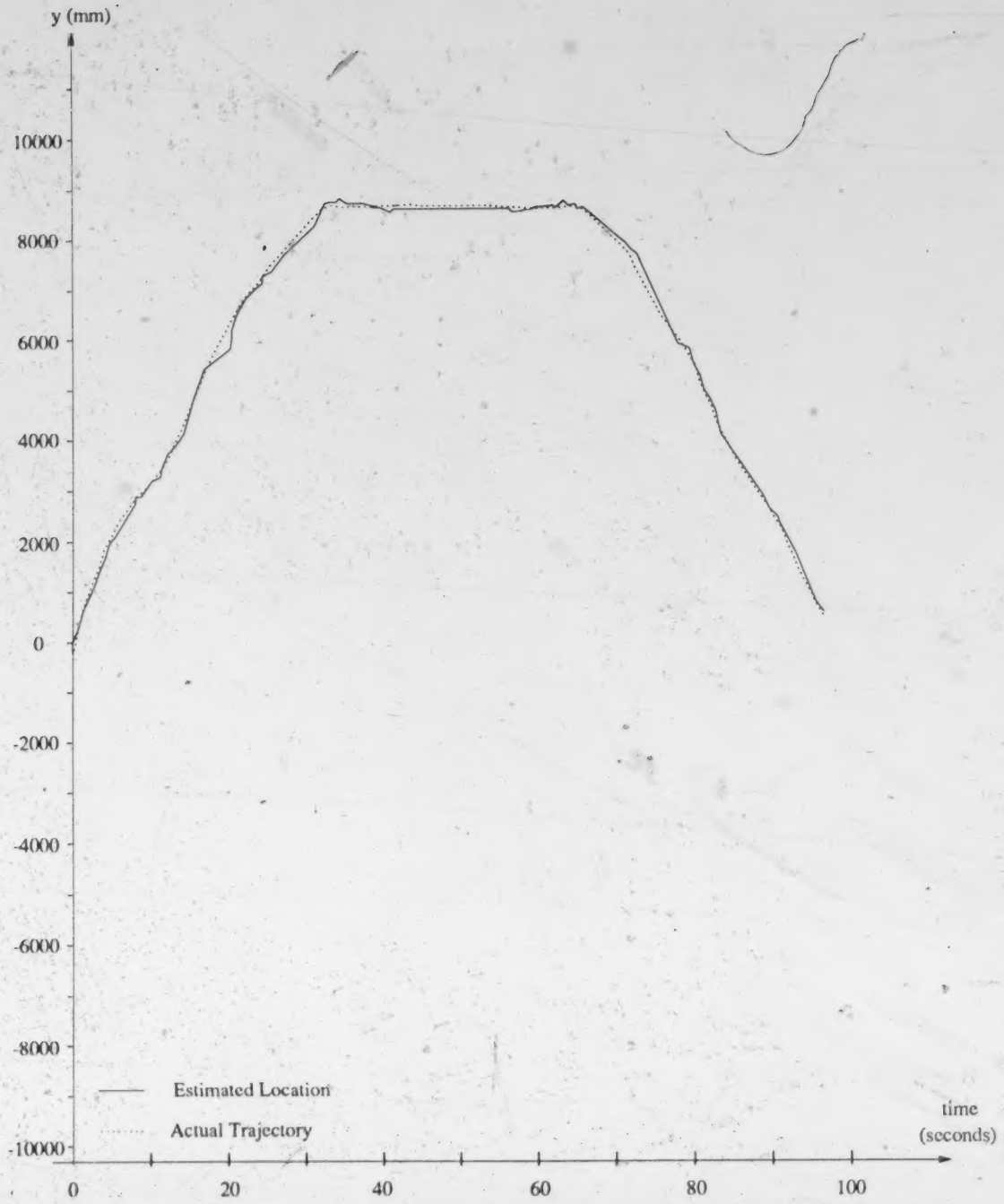

Figure 5.19: y coordinate of the actual and the estimated location

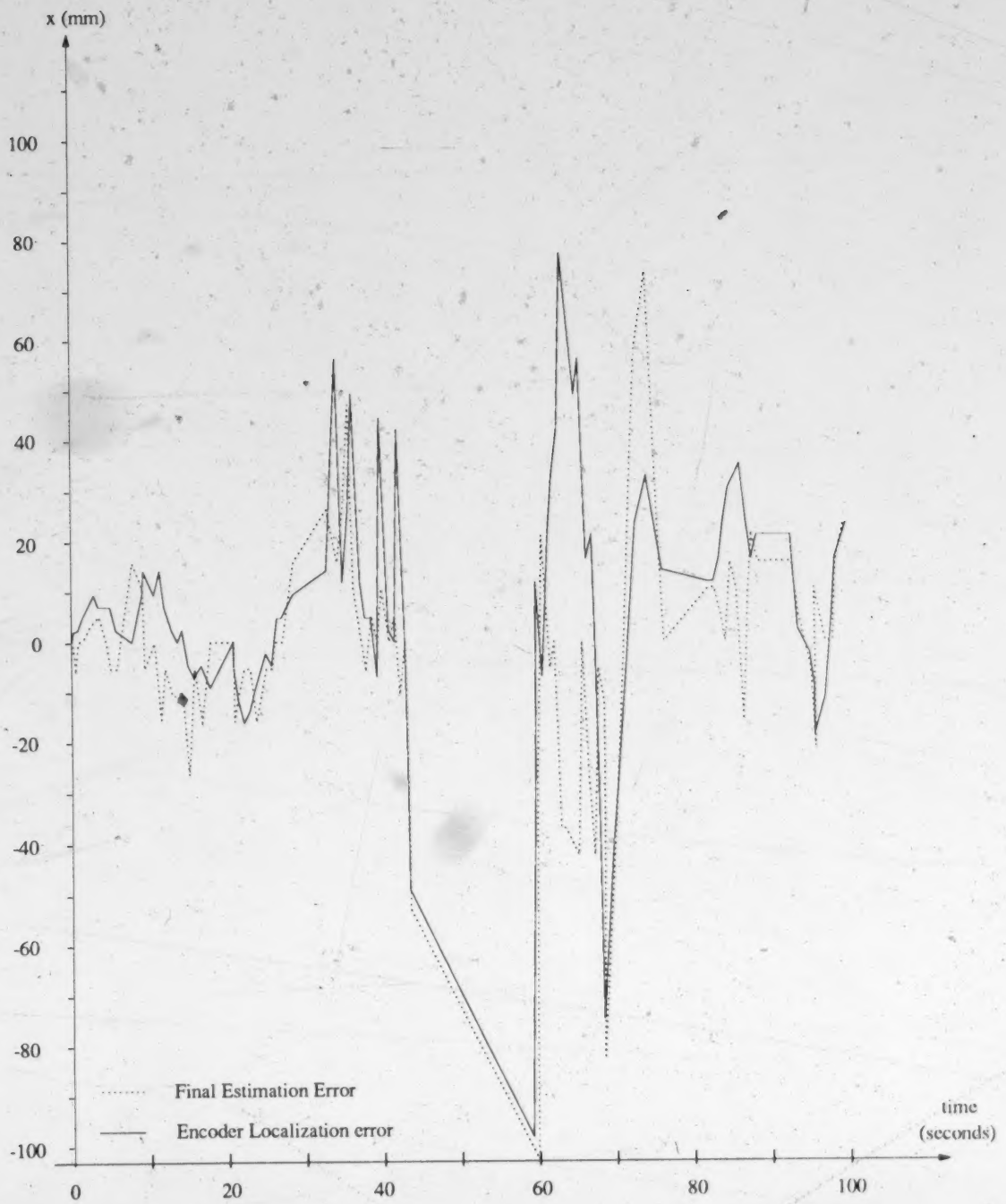

Figure 5.20: Error in Encoder localization and final estimation in x coordinates

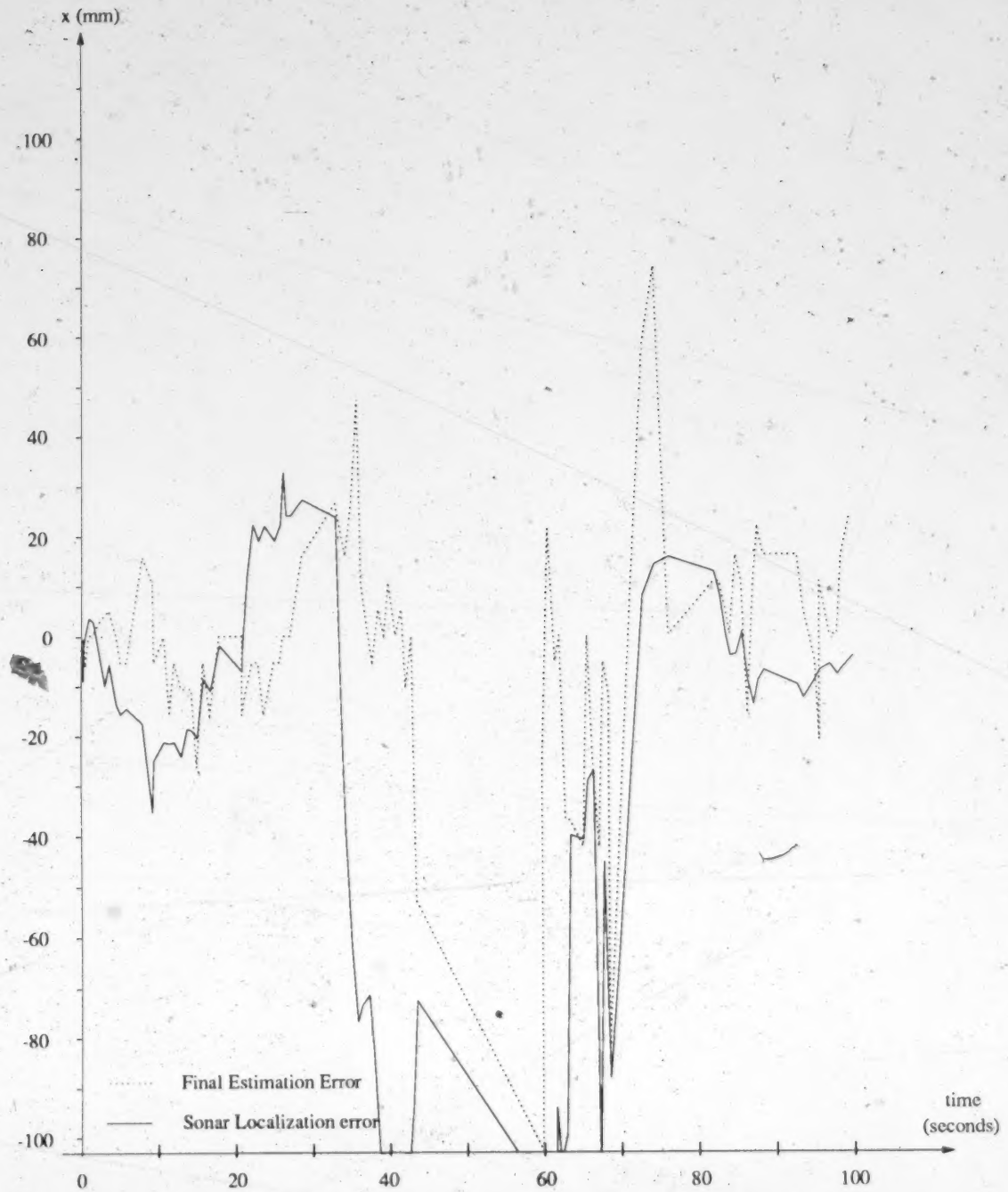

Figure 5.21: Error in Sonar localization and final estimation in x coordinates

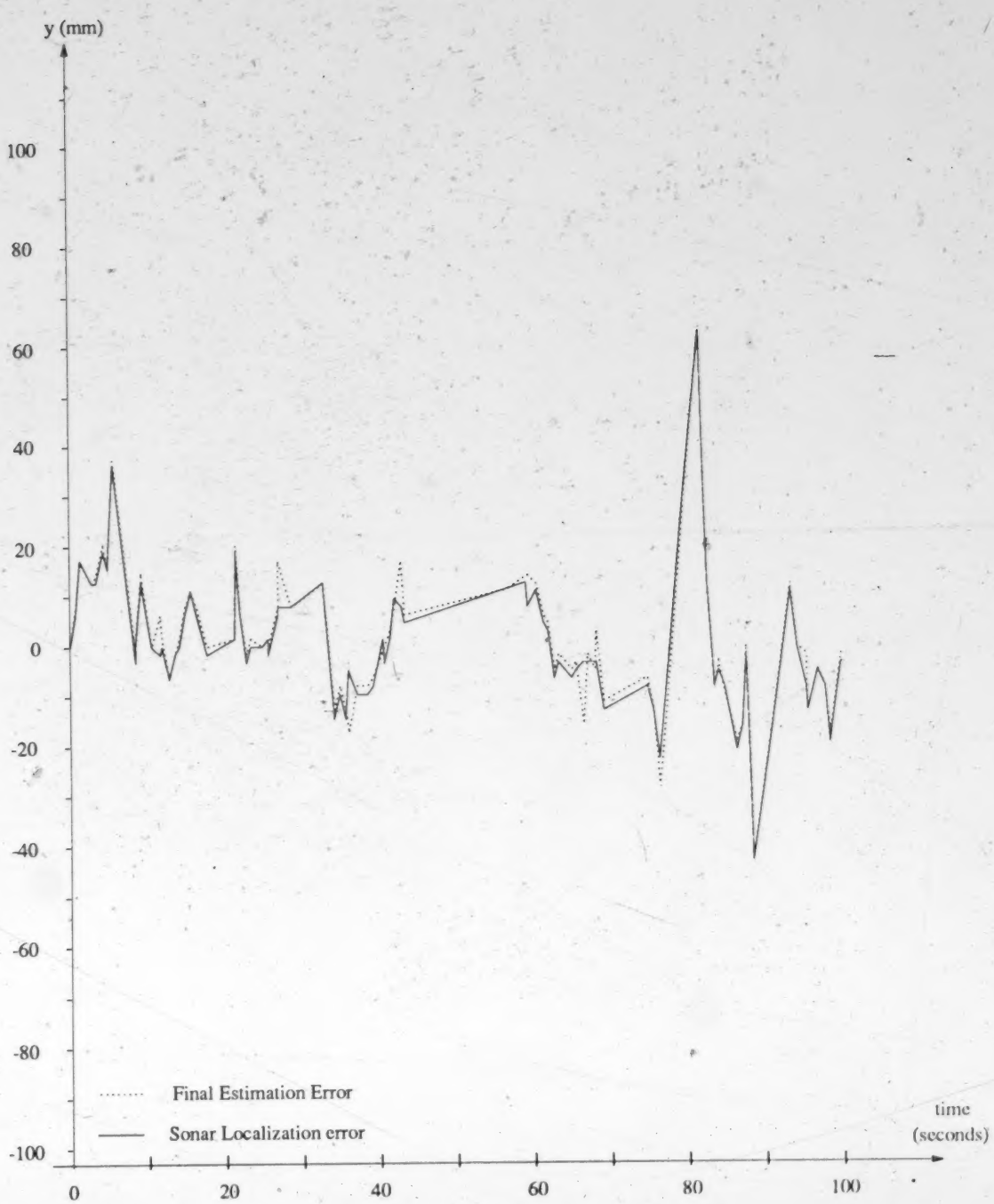

Figure 5.22: Error in Encoder localization and final estimation in y coordinates

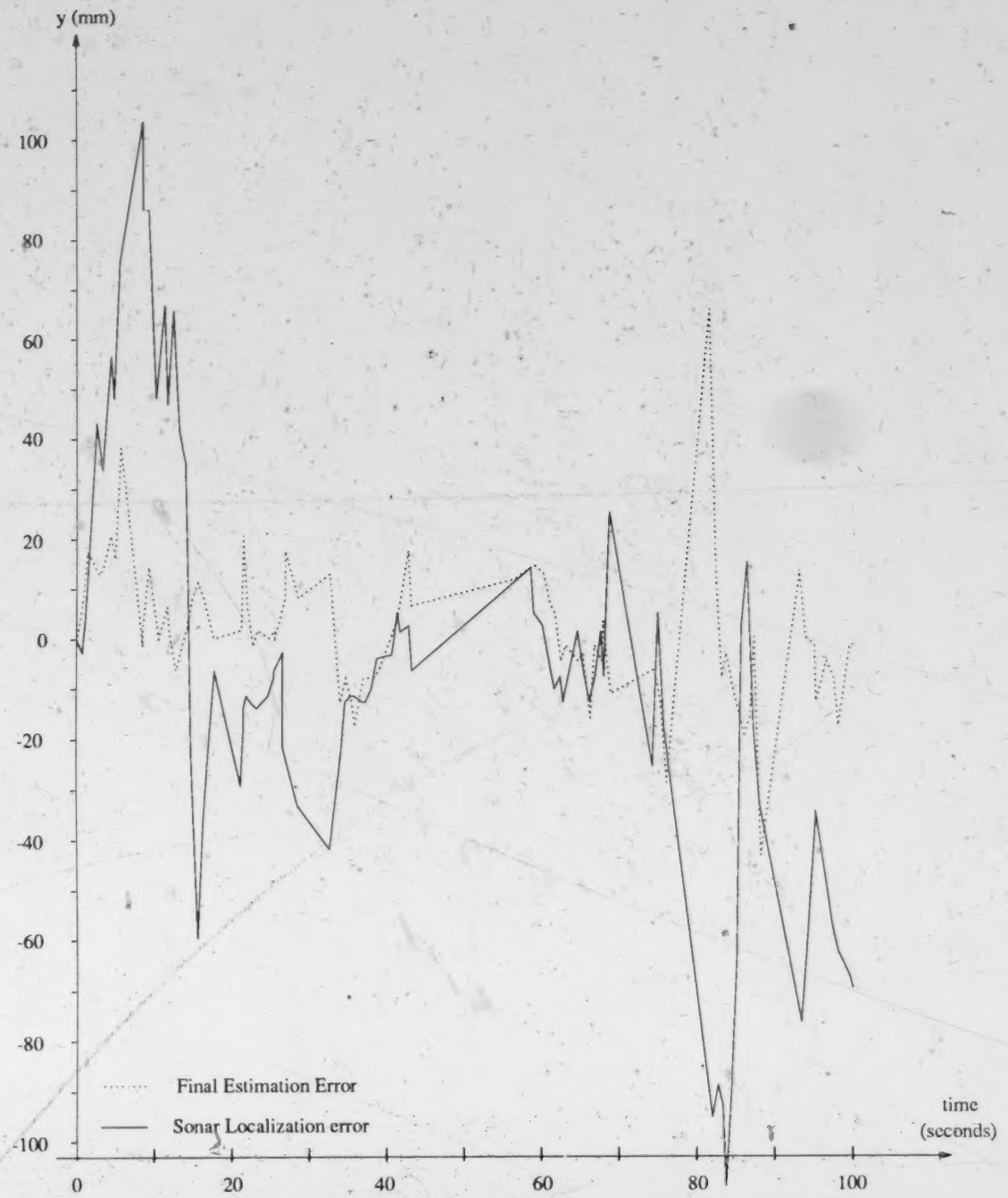

Figure 5.23: Error in Sonar localization and final estimation in y coordinates

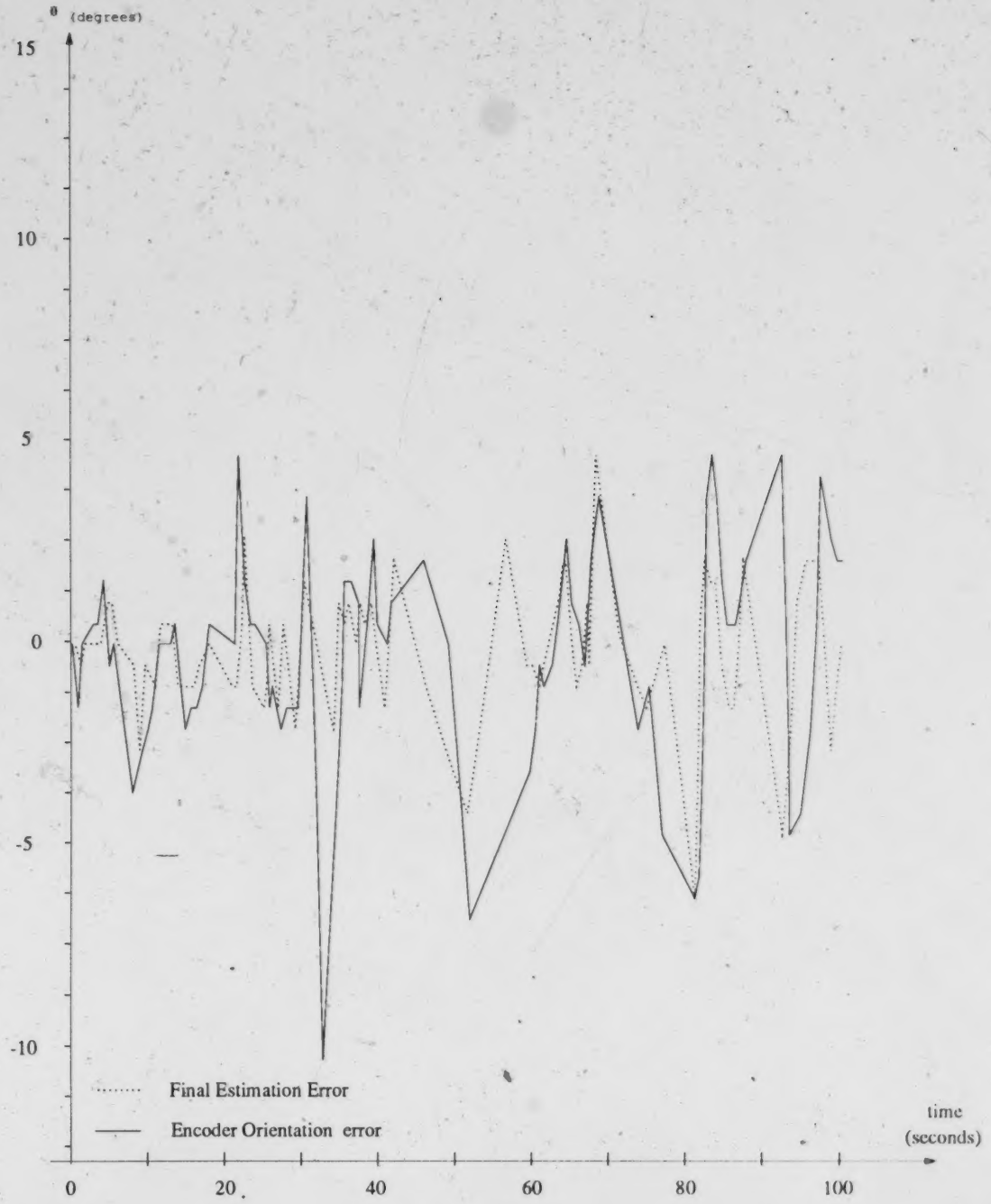

Figure 5.24: Error in Encoder localization and final estimation in orientation

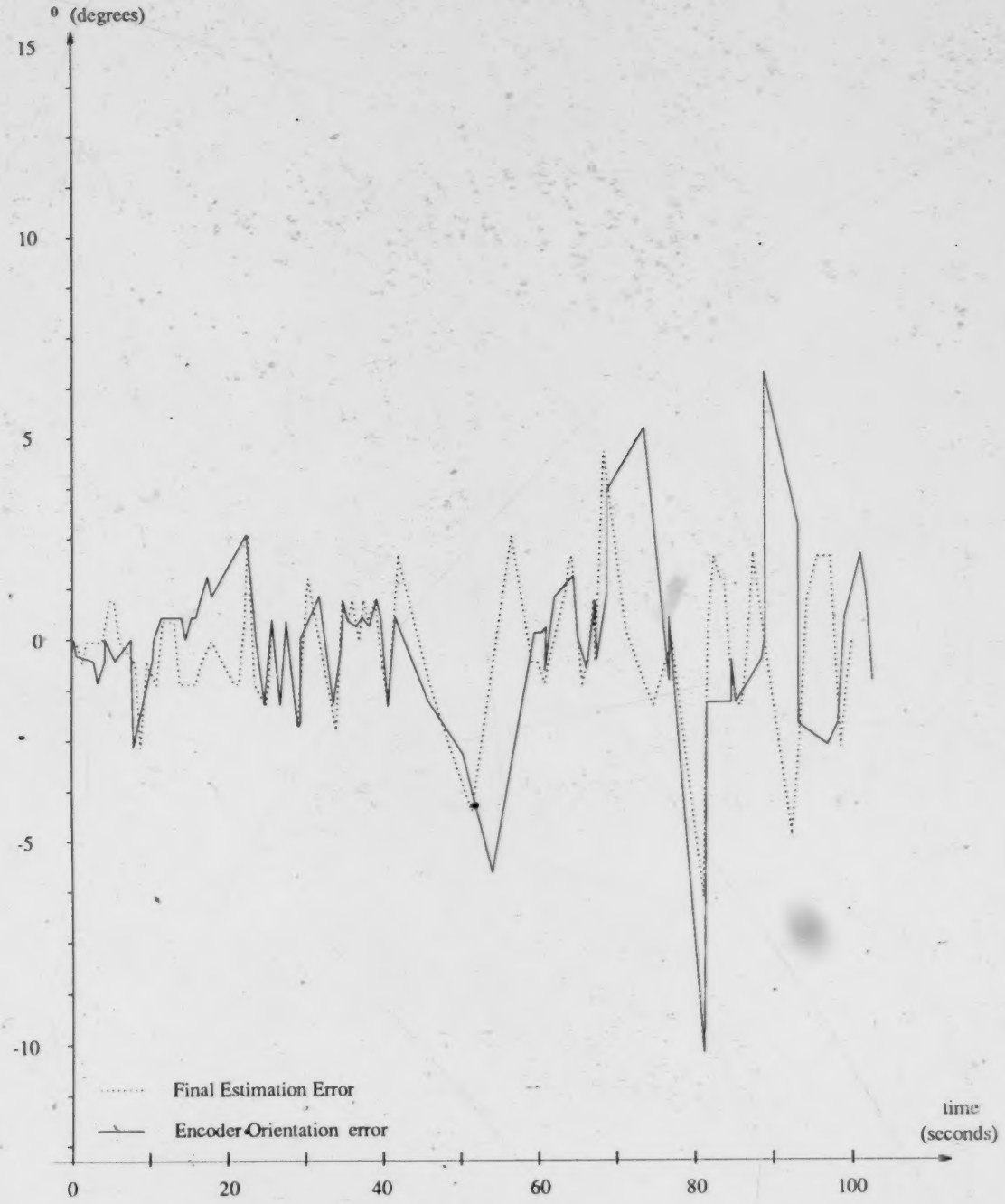

Figure 5.25: Error in Sonar localization and final estimation in orientation

Chapter 6

Conclusion

6.1 Summary Discussion

We have presented an autonomous mobile navigation system in a previously known environment. There are two key elements of the system: a position estimation algorithm and a real-time trajectory generation and control method.

The position estimation algorithm is composed of three components: the world model, the sensor models and the data fusion method. The world model provides a precise description of the environment's geometric features. The sensor models describe the behavior of the sensors and the uncertainties related to them. The fusion method is a general scheme which combines multiple sensory inputs into a single estimate based on their uncertainties.

The methods introduced for sensor localization and data fusion are based on a probabilistic description of the uncertainty involved in sensor readings. Heuristic and statistical decision making criteria are used to cluster and integrate sensory information in a consistent manner. The localization method is sensitive to recent characteristics of the environment. A data fusion scheme integrates location informa-

tion from the sensory systems. A Bayesian maximum likelihood estimation method is used to obtain the final position estimate which provides a general scheme for sensory data fusion.

The trajectory generation method introduced in this thesis has been developed on the basis of previously introduced methods. The method uses linear and constant curvatured segments to provide an efficient and practically continuous trajectory. The continuity of the trajectory maintains smooth navigation. It is computationally simple, so a real-time system can use the method without a burden of computation. In addition, the path generation functions are not bound to certain types of actuation mechanisms. They are designed to be flexible so that they can be adopted by other systems.

Unlike many other previous attempts [Elfes, 1986] [Crowley, 1985], we presented a navigation method where the perception and trajectory control are done during a continuous motion of the robot. The resulting system is able to maintain a robust position estimate while it is navigating in an a priori environment.

We have experimentally verified the validity of the introduced autonomous navigation method on our mobile robot. The implemented application tests the localization, sensory fusion and trajectory generation and control techniques using a two wheeled robot with 16 sonar proximity sensors and 2 optical differential encoders mounted on the wheels. The robot navigates successfully through the corridors and it reacts to the deviations from desired paths.

6.2 Future Research

The system we developed will act as a basis for future development. The planned enhancements on top of the system include: using machine vision for recognition of

objects including landmarks of geometric features of the environment for position estimation, reducing sensor uncertainty, detecting physical reflection patterns from sonar range data, and improvements on the hardware and the computational power of the system.

Our localization method facilitates redundant sensors to be introduced into the system. Vision (passive sensing) is one of the first candidates for future enhancements of the robot. Machine vision is useful for locating and examining objects in disordered environments. A reliable vision system for a mobile robot requires evaluating image sequences in order to detect and differentiate between objects. Several position estimation techniques using correspondences between landmarks in the environment have been developed in the literature [Krotkov, 1989], [Sugihara, 1987]. These methods need to be further developed to consider the uncertainties in noisy rays in subsequent images. The developed uncertainty model can be used to introduce the vision system into our localization scheme.

The uncertainty in sensor readings can be further reduced using an optimum linear filter. A powerful technique for real time estimation of dynamic systems is the Kalman filter [Sage and Melsa, 1971]. This formulation allows the integration of information over time, and is robust with respect to sensor noise. The sensor models must be further analyzed to compute the prediction and update phases of the Kalman gain factor. Some applications have been done using this method [Mathies *et al.*, 1988], [Crowley, 1989b]. However, more research for a general solution for non-lateral motion and for non-biased estimates in unstructured environments are required.

The physical characteristics of sonar reflections can be used to recognize different objects in the environments. Some physically based simulation models have been proposed for acoustic sensors [Miwa and Kuc, 1986], [Kuc and Siegel, 1987]. These methods may be further elaborated for pattern recognition to help positioning of the robot.

A significant problem with the sensor system is not knowing the exact location of the robot when an echo is received. This may be improved by modifying the proximity system hardware so that it will issue a relative time or position parameter when an echo is received. This improvement will lead to a significant improvement in the precision of the sonar localization algorithm.

Further improvements of the computational power for a mobile robot is an essential requisite in mobile robotics. New improvements and techniques are rapidly being developed in the field. With the introduction of these new techniques, it is inevitable that the burden of computation will reach a high level of complexity. This will require a powerful multi-processing, real-time and onboard computational power. Currently a MC68030 VME board [Mot, 1989] is being planned to be mounted on the robot in order to take over the computational tasks. The MC68030 board will not only improve the computational power of the robot, but it will also provide full autonomy by eliminating the serial connections between the robot and the host computer. However, the development of an operating system kernel will be required.

Bibliography

[Brooks and Lozano-Perez, 1983] R.A. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Journal on Systems, Man, and Cybernetics*, 2(3), 1983.

[Brooks, 1988] R. Brooks. *Tutorial Notes # 7, Autonomous Mobile Robots*. CSCSI88, Edmonton, 1988.

[Chatila and Giralt, 1986] R. Chatila and G. Giralt. Task and path planning for mobile robots. In *NATO ARW on Machine Intelligence and Knowledge Engineering*, Maratea,Italy, May 1986.

[Crowley, 1985] J. Crowley. Navigation for an intelligent mobile robot. *IEEE Journal of Robotics and Automation*, 1(1):31–41, 1985.

[Crowley, 1989a] J. Crowley. Asynchronous control of orientation and displacement in a robotic vehicle. In *IEEE International Conference on Robotics and Automation*, pages 1277–1282, 1989.

[Crowley, 1989b] J. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *IEEE International Conference on Robotics and Automation*, pages 674–680, 1989.

[Drake *et al.*, 1985] K.C. Drake, E.S. McVey, and R.M. Inigo. Sensing error for a mobile robot using line navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):485–490, July 1985.

[Drumheller, 1987] M. Drumheller. Mobile robot localization using sonar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):325–332, March 1987.

[Duda and Hart, 1973] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons Inc., 1973.

[Durrant-Whyte, 1987] H.F. Durrant-Whyte. Consistent integration and propagation of disparate sensor observations. *International Journal of Robotics Research*, 6(3):3–24, 1987.

[Elfes, 1986] A. Elfes. A sonar based mapping and navigation system. In *IEEE 1986 International Conference on Robotics and Automation*, pages 1151–1156, 1986.

[Faverjon and Tournossoud, 1987] B. Faverjon and P. Tournossoud. A local based approach for path planning of manipulators with a high number of freedom. In *IEEE, International Conference on Robotics and Automation*, Raleigh, 1987.

[Giralt *et al.*, 1979] G. Giralt, R. Sobek, and R. Chatila. A multi level planning and navigation system for a mobile robot. In *Proceedings of the 6th IJCAI. Tokyo, Japan*, pages 13–20, October 1979.

[Giralt *et al.*, 1987] G. Giralt, R. Chatila, and M. Varsoff. An integrated navigation and motion control system for autonomous multisensory mobile robots. In Brady M and R. Paul, editors, *Robotics Research The First International Symposium.* MIT Press, 1987.

[Giralt, 1984] G. Giralt. Mobile robots. In Brady et.al., editor, *Robotics and Artificial intelligence*, pages 365–393. Springer Verlag, 1984.

[Hackett and Shah, 1990] J.K. Hackett and M. Shah. Multi sensor fusion, a perspective. In *IEEE 1990 International Conference on Robotics and Automation*, 1990.

[Hirose, 1984] E. Hirose. Adaptive gait control of a quadroped walking machine. In *The First International Symposium on Robotics Research*, pages 253–277, 1984.

[Hoppen *et al.*, 1990] P. Hoppen, T. Kierman, and E. von Puttkamer. Laser-radar based mapping and navigation for an autonomous mobile robot. In *IEEE 1990 International Conference on Robotics and Automation*, pages 948–953, 1990.

[Hu and Stockman, 1986] G. Hu and G. Stockman. 3-d scene analysis via fusion of light stripped image and intensity image. In F.C.A. Groen and L.O. Hertzberger, editors, *1987 Workshop on Spatial Reasoning and Multi Sensor Fusion*, pages 138–147, October 1986.

[Kanade, 1987] T. Kanade. *Three Dimensional Machine Vision*. Klower Academic Publishers Inc., 1987.

[Kanayama and Miyake, 1986] Y. Kanayama and N. Miyake. Trajectory generation for mobile robots. *Robotics Research 3*, pages 333–340, 1986.

[Kent *et al.*, 1986] E.W. Kent, M.O. Scheiner, and T. Hong. Building representations from fusion of multiple views. In *IEEE, International Conference on Robotics and Automation*, pages 1634–1639, April 1986.

[Khatib, 1986] O. Khatib. Real time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 1(5), 1986.

[Kriegman *et al.*, 1987] D. Kriegman, E. Triendl, and T. O. Binford. A mobile robot: Sensing planning and locomotion. In *IEEE 1987 International Conference on Robotics and Automation*, pages 402–408, 1987.

[Krotkov, 1989] E. Krotkov. Mobile robot navigation using a single image. In *IEEE 1989 International Conference on Robotics and Automation*, pages 978–983, 1989.

[Kuc and Barshan, 1989] R. Kuc and B. Barshan. Navigating vehicles through an unstructured environment using sonar. In *IEEE 1989 International Conference on Robotics and Automation*, pages 1422–1426, 1989.

[Kuc and Siegel, 1987] R. Kuc and M.W. Siegel. Physically based simulation model for acoustic sensor robot navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):766–778, 1987.

[Lao *et al.*, 1986] Z.L. Lao, J. Ronning, and E.L. Hall. Omnidirectional vision navigation integrating beacon recognition with positioning. In F.C.A. Groen and L.O. Hertzberger, editors, *Proceedings of the SPIE Conference on Mobile Robots*, pages 213–221, October 1986.

[Laumond, 1987] J.P. Laumond. Feasible trajectories for mobile robots with kinematic and environment constraints. In F.C.A. Groen and L.O. Hertzberger, editors, *Intelligent Autonomous Systems*. North Holland, 1987.

[Laumond, 1989] J.P. Laumond. Trajectory planning and motion control for mobile robots. In J.D. Boissonnat and J.P. Laumond, editors, *Geometry and Robotics*, pages 133–149. Springer Verlag, 1989.

[Levitt *et al.*, 1987] T.S. Levitt, D.T. Lawton, D.M. Chalberg, and P.C. Nelson. Qualitative landmark based path planning and following. In *Proceedings of the AAAI Conference.Seattle, Washington*, pages 689–694, July 1987.

[Lin *et al.*, 1989] L. Lin, T.M. Mitchell, A. Philips, and R. Simmons. A case study in robot exploration. Technical Report TR-89-1, Carnegie Mellon university, The Robotics Institute, 1989.

[Luo *et al.*, 1988] R.C. Luo, M. Lin, R.S. Scharp, and P.E. Wessel. Object recognition using tactile image array sensors. *IEEE Journal of Robotics Research*, pages 568–573, 1988.

[Maeda, 1985] Y. Maeda. Prototype of multifunctional robot vehicle. In *ICAR Conference, Tokyo*, pages 421–428, 1985.

[Mathies and Elfes, 1988] L. Mathies and A. Elfes. Integration of sensor and stereo range data using a grid based representation. In *IEEE 1988 International Conference on Robotics and Automation*, pages 727–733, 1988.

[Mathies *et al.*, 1988] L. Mathies, R. Szeliski, and T. Kanade. Kalman filter based algorithms for estimating depth from image sequences. Technical Report TR-88-1, Carnegie Mellon university, The Robotics Institute, 1988.

[Miller, 1984] D. Miller. Two dimensional mobile robot positioning using onboard sonar. In *IEEE 1984 International Conference on Robotics and Automation*, pages 766–778, 1984.

[Miwa and Kuc, 1986] H. Miwa and R. Kuc. A computer model for simulating reflected ultrasonic signals. *Journal of Acoustic Society*, 80(3):951–954, September 1986.

[Moravec and Elfes, 1986] H.P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *IEEE 1986 International Conference on Robotics and Automation*, 1986.

[Mot, 1989] Motorola Corporation. *Motorola MVME147 Monoboard Computer*, 3 edition, 1989.

[Perez and Rouchy, 1987] J.C. Perez and I. Rouchy. Increasing autonomy of assembly robots. In F.C.A. Groen and L.O. Hertzberger, editors, *Intelligent Autonomous Systems*. North Holland, 1987.

[Raibert, 1984] M.H. Raibert. Robots that walk. In Brady et.al., editor, *Robotics and Artificial intelligence*, pages 345–364. Springer Verlag, 1984.

[Rembold and Levi, 1987] U. Rembold and P. Levi. Sensors and control. In F.C.A. Groen and L.O. Hertzberger, editors, *Intelligent Autonomous Systems*, pages 79 95. North Holland, 1987.

[Rodger and Browse, 1986] J.C. Rodger and R.A. Browse. An object based representation for multisensory robotic perception. In F.C.A. Groen and L.O. Hertzberger, editors, *1987 Workshop on Spatial Reasoning and Multi Sensor Fusion*, pages 213–221, October 1986.

[Sage and Melsa, 1971] A.P. Sage and J.L. Melsa. *Estimation Theory with Applications to Communications and Control.* McGraw-Hill Book Company, 1971.

[Schmidt, 1971] R.A. Schmidt. *A study of a real-time control of a computer driven vehicle.* PhD thesis, Stanford University, 1971.

[Schwartz and Scharir., 1983] J.T. Schwartz and M. Scharir. On the piano mover: the case of a two dimensional rigid polynomial body moving amidst polygonal barriers. *Communication on Pure and Applied Math*, 36, 1983.

[Silk, 1983] M.G. Silk. An extended model of the ultrasonic transducer. *Journal of Physics, E*, 16:879–887, 1983.

[Smith and Cheeseman, 1987] R Smith and P. Cheeseman. On the representation of the estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, 1987.

[Stefik, 1985] M. Stefik. Strategic computing at DARPA. *Communications ACM*, 28(7):690–704, 1985.

[Sugano, 1985] S. Sugano. Limb control of the robot musician. In *ICAR Conference, Tokyo*, pages 471–476, 1985.

[Sugihara, 1987] K. Sugihara. Location of a robot using visual information. In *The Fourth International Symposium on Robotics Research*, pages 81–88, 1987.

[Sun, 1989] Sun Microsystems. *Unix Manuals, C library Routines*, a-17 edition, 1989.

[Thomas, 1987] J.B. Thomas. *Applied Probability and Random Processes.* John Wiley and Sons Inc., 1987.

[Thorpe and Kanade, 1986] C. Thorpe and T. Kanade. Vision and Navigating for the CMU Navlab. In *SPIE Conference on Mobile Robots. Cambridge, Mass*, pages 260–266, October 1986.

[Tournossoud, 1988] P. Tournossoud. Motion planning for a mobile robot with a kinematic constraint. In *IEEE International Conference on Robotics and Automation*, 1988.

[Tra, 1989a] Transitions Research Corporation. *Proximity Subsystem User's Manual*, release 4.6e edition, 1989.

[Tra, 1989b] Transitions Research Corporation. *TRC Labmate, Autonomous Mobile Robot Base*, release 5.4f edition, 1989.

[Tsumura *et al.*, 1981] T. Tsumura, N. Fujiwara, T. Shirikawa, and N. Hashimoto. An experimental system for automatic guidance of robot, a vehicle following the road stored in memory. In 11^{th} *International Symposium on Industrial Robots*, pages 187–193, October 1981.

[Wallace *et al.*, 1986] R. Wallace, K. Matsuzaki, Y. Goto, J. Crisman, J. Webb, and T. Kanade. Progress in robot road following. *IEEE International Conference on Robotics and Automation*, 1986.

[Zhimin and Dongying, 1985] S. Zhimin and G. Dongying. Kinematics of six legged vehicle on irregular terrain. In *ICAR Conference, Tokyo*, pages 389–396, 1985.